Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2011

A. Melnikov, Ed.
Isode Ltd

March 7, 2011

Internationalized Email Addresses in X.509 certificates
draft-ietf-pkix-eai-addresses-00

Abstract

This document defines a new name form for inclusion in the otherName field of an X.509 Subject Alternative Name extension that allows a certificate subject to be associated with an Internationalized Email Address.

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 http://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 September 8, 2011.

Table of Contents

1. Introduction

[RFC5280] defines rfc822Name subjectAltName choice for representing [RFC5322] email addresses. This form is restricted to a subset of US-ASCII characters and thus can't be used to represent Internationalized Email addresses [I-D.ietf-eai-rfc5336bis].

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The formal syntax use the Augmented Backus-Naur Form (ABNF) [RFC5234] notation.

3. Name Definitions

This section defines the eaiName name as a form of otherName from the GeneralName structure in SubjectAltName defined in [RFC5280].

```
id-on-eaiAddr OBJECT IDENTIFIER ::= { id-on XXX }

eaiName ::= UTF8String (SIZE (1..MAX))
```

When the subjectAltName extension contains an Internationalized Email address, the address MUST be stored in the eaiName name form of otherName. The format of an eaiName is a <eaiMailbox> as defined below. A eaiMailbox has the form "Local-part@Domain". Note that a eaiMailbox has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

```
eaiMailbox = uLocal-part "@" uDomain
```

uLocal-part and uDomain are defined in [I-D.ietf-eai-rfc5336bis].

4. Matching of Internationalized Email Addresses in X.509 certificates

The <uLocal-part> part of an Internationalized email address is in UTF-8 and need to be compared octet for octet.

The <uDomain> can contain either IDN domain or an ASCII Compatible Encoding (ACE) format. When comparing two <uDomain>s both MUST be converted to the ACE form as described in section 7.2 of [RFC5280].

5. IANA Considerations

[[anchor6: Just need a new OID.]]

6. Security Considerations

[[anchor7: TBD]]

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[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, May 2008.

[I-D.ietf-eai-rfc5336bis] Yao, J. and W. Mao, "SMTP extension for internationalized email address", draft-ietf-eai-rfc5336bis-08 (work in progress), March 2011.

7.2. Informative References

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

Appendix A. Acknowledgements

Thank you to Magnus Nystrom for motivating this document.

Author's Address

Alexey Melnikov (editor)
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

Network Working Group
Internet-Draft
Intended status: Proposed Standard
Updates: 2560 (once approved)
Expires: September 12, 2011

S. Santesson
3xA Security
P. Hallam-Baker
Default Deny Security

March 11, 2011

Online Certificate Status Protocol Algorithm Agility
draft-ietf-pkix-ocspagility-10

Status of this Memo

Copyright and License Notice

Abstract

The Online Certificate Status Protocol (OSCP) [RFC2560] requires server responses to be signed but does not specify a mechanism for selecting the signature algorithm to be used. This may lead to avoidable interoperability failures in contexts where multiple signature algorithms are in use. This document specifies rules for server signature algorithm selection and an extension that allows a client to advise a server that specific signature algorithms are supported.

Table of Contents

1 Introduction

1.1 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2 OCSP Algorithm Agility Requirements

The Online Certificate Status Protocol (OCSP) [RFC2560] defines a protocol for obtaining certificate status information from an online service. An OCSP Responder may or may not be issued an OCSP Responder certificate by the certification authority (CA) that issued the certificate whose status is being queried. An OCSP Responder may provide pre-signed OCSP responses or may sign responses when queried.

RFC 2560 [RFC2560] specifies a means for an OCSP responder to indicate the signature and digest algorithms used in a response but not how those algorithms are specified. The only algorithm requirements established by that protocol specification are that the OCSP client SHALL support the DSA algorithm sig-alg-oid specified in section 7.2.2 of [RFC2459] and SHOULD be capable of processing RSA signatures as specified in section 7.2.1 of [RFC2459]. The only requirement placed on responders by RFC 2560 is that they SHALL support the SHA1 hashing algorithm.

Since algorithms other than the mandatory to implement algorithms are Allowed, and since a client currently has no mechanism to indicate it’s algorithm preferences, there is always a risk that a server choosing a non-mandatory algorithm, will generate a response that the client may not support.

While an OCSP Responder may apply rules for algorithm selection, e.g., using the signature algorithm employed by the CA for signing CRLs and certificates, such rules may fail in common situations:

- The algorithm used to sign the CRLs and certificates may not be consistent with key pair being used by the OCSP Responder to sign responses.
- A request for an unknown certificate provides no basis for a responder to select from among multiple algorithm options.

The last criterion cannot be resolved through the information available from in-band signaling using the RFC 2560 [RFC2560] protocol, without modifying the protocol.

In addition, an OCSP Responder may wish to employ different signature algorithms than the one used by the CA to sign certificates and CRLs for several reasons:

- The responder may employ an algorithm for certificate status response that is less computationally demanding than for signing the certificate itself.
- An implementation may wish to guard against the possibility of a compromise resulting from a signature algorithm compromise by employing two separate signature algorithms.

This document describes:

- A mechanism that allows a client to indicate the set of preferred signature algorithms.
- Rules for signature algorithm selection that maximizes the probability of successful operation in the case that no supported preferred algorithm(s) are specified.

3 Updates to Mandatory and Optional Cryptographic Algorithms

Section 4.3 "Mandatory and Optional Cryptographic Algorithms" of RFC 2560 [RFC2560] is updated as follows:

OLD: Clients that request OCSP services SHALL be capable of processing responses signed used DSA keys identified by the DSA sig-alg-oid specified in section 7.2.2 of [RFC2459]. Clients SHOULD also be capable of processing RSA signatures as specified in section 7.2.1 of [RFC2459]. OCSP responders SHALL support the SHA1 hashing algorithm.

NEW: Clients that request OCSP services SHALL be capable of processing responses signed using RSA with SHA-1 (identified by sha1WithRSAEncryption OID specified in [RFC3279]) and RSA with SHA-256 (identified by sha256WithRSAEncryption OID specified in [RFC4055]). Clients SHOULD also be capable of processing responses signed using DSA keys (identified by the id-dsa-with-sha1 OID specified in [RFC3279]). Clients MAY support other algorithms.

4 Client Indication of Preferred Signature Algorithms

A client MAY declare a preferred set of algorithms in a request by including a preferred signature algorithms extension in requestExtensions of the OCSPRequest [RFC2560].

```
id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

PreferredSignatureAlgorithms ::= SEQUENCE OF
                                 PreferredSignatureAlgorithm

PreferredSignatureAlgorithm ::= SEQUENCE {
   sigIdentifier        AlgorithmIdentifier,
   pubKeyAlgIdentifier  SMIMECapability OPTIONAL
   }
```

The syntax of AlgorithmIdentifier is defined in section 4.1.1.2 of RFC 5280 [RFC5280] The syntax of SMIMECapability is defined in RFC 5751 [RFC5751]

sigIdentifier specifies the signature algorithm the client prefers, e.g. algorithm=ecdsa-with-sha256. Parameters are absent for most common signature algorithms.

pubKeyAlgIdentifier specifies the subject public key algorithm identifier the client prefers in the server's certificate used to validate the OCSP response. e.g. algorithm=id-ecPublicKey and parameters= secp256r1.

pubKeyAlgIdentifier is OPTIONAL and provides means to specify parameters necessary to distinguish among different usages of a particular algorithm, e.g. it may be used by the client to specify what curve it supports for a given elliptic curve algorithm.

The client MUST support each of the specified preferred signature algorithms and the client MUST specify the algorithms in the order of preference, from the most preferred to the least preferred.

Section 5 of this document describes how a server selects an algorithm for signing OCSP responses to the requesting client.

5 Responder Signature Algorithm Selection

RFC 2560 [RFC2560] does not specify a mechanism for deciding the signature algorithm to be used in an OCSP response. As previously noted this does not provide a sufficient degree of certainty as to the algorithm selected to facilitate interoperability.

5.1 Dynamic Response

A responder MAY maximize the potential for ensuring interoperability by selecting a supported signature algorithm using the following order of precedence, as long as the selected algorithm meets all security requirements of the OCSP responder, where the first method has the highest precedence:

1. Select an algorithm specified as a preferred signing algorithm in the client request

2. Select the signing algorithm used to sign a certificate revocation list (CRL) issued by the certificate issuer providing status information for the certificate specified by CertID

3. Select the signing algorithm used to sign the OCSPRequest

4. Select a signature algorithm that has been advertised as being the default signature algorithm for the signing service using an out of band mechanism

5. Select a mandatory or recommended signing algorithm specified for the version of the OCSP protocol in use

A responder SHOULD always apply the lowest numbered selection mechanism that results in the selection of a known and supported algorithm that meets the responder’s criteria for cryptographic algorithm strength.

5.2 Static Response

For purposes of efficiency, an OCSP responder is permitted to generate static responses in advance of a request. The case may not permit the responder to make use of the client request data during the response generation, however the responder SHOULD still use the client request data during the selection of the pre-generated response to be returned. Responders MAY use the historical client requests as part of the input to the decisions of what different algorithms should be used to sign the pre-generated responses.

6 Acknowledgements

The authors acknowledges Santosh Chokhani for the helpful comments made on earlier drafts, Sean Turner for proposing the syntax for algorithm identifiers, Jim Schaad for providing and testing the ASN.1 module in Annex A and Stephen Kent for valuable review and input.

7 IANA Considerations

This document requires no actions by IANA.

8 Security Considerations

The mechanism used to choose the response signing algorithm MUST be considered to be sufficiently secure against cryptanalytic attack for the intended application.

In most applications it is sufficient for the signing algorithm to be at least as secure as the signing algorithm used to sign the original certificate whose status is being queried. This criteria may not hold in long term archival applications however in which the status of a certificate is being queried for a date in the distant past, long after the signing algorithm has ceased being considered trustworthy.

8.1 Use of insecure algorithms

It is not always possible for a responder to generate a response that the client is expected to understand and that meets contemporary standards for cryptographic security. In such cases an OCSP responder operator MUST balance the risk of employing a compromised security solution and the cost of mandating an upgrade, including the risk that the alternative chosen by end users will offer even less security or no security.

In archival applications it is quite possible that an OCSP responder might be asked to report the validity of a certificate on a date in the distant past. Such a certificate might employ a signing method that is no longer considered acceptably secure. In such circumstances the responder MUST NOT generate a signature using a signing mechanism that is not considered acceptably secure.

A client MUST accept any signing algorithm in a response that it specified as a preferred signing algorithm in the request. It follows therefore that a client MUST NOT specify as a preferred signing algorithm any algorithm that is either not supported or not

considered acceptably secure.

8.2 Man in the Middle Downgrade Attack

The mechanism to support client indication of preferred signature algorithms is not protected against a man in the middle downgrade attack. This constraint is not considered to be a significant security concern since the OCSP Responder MUST NOT sign OCSP Responses using weak algorithms even if requested by the client. In addition, the client can reject OCSP responses that do not meet its own criteria for acceptable cryptographic security no matter what mechanism is used to determine the signing algorithm of the response.

8.3. Denial of Service Attack

Algorithm agility mechanisms defined in this document introduces a slightly increased attack surface for Denial-of-Service attacks where the client request is altered to require algorithms that are not supported by the server. Denial-of-Service considerations from RFC 4732 [RFC4732] are relevant for this document.

9 References

9.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[RFC3279] W. Polk, R. Housley, L. Bassham, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC4055] J. Schaad, B. Kaliski, R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

[RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

9.2 Informative References

[RFC2459] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure - Certificate and CRL Profile", January 1999

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.

Appendix A - ASN.1 Modules

A.1 ASN.1 Module

```
 OCSP-AGILITY-2009 { iso(1) identified-organization(3) dod(6)
     internet(1)  security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-ocsp-agility-2009-93(66) }

 DEFINITIONS EXPLICIT TAGS ::=
 BEGIN

   EXPORTS ALL;   -- export all items from this module
   IMPORTS

   id-pkix-ocsp
     FROM OCSP-2009  -- From OCSP [RFC2560]
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp-02(48) }

   AlgorithmIdentifier{}, SMIMECapability{}, SIGNATURE-ALGORITHM,
   PUBLIC-KEY
     FROM AlgorithmInformation-2009 -- From [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-algorithmInformation-02(58) }

   EXTENSION
     FROM PKIX-CommonTypes-2009 -- From [RFC5912]
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)} ;

    --  Add re-preferred-signature-algorithms to the set of extensions
    --  for TBSRequest.requestExtensions

   re-preferred-signature-algorithms EXTENSION ::= {
      SYNTAX PreferredSignatureAlgorithms
      IDENTIFIED BY id-pkix-ocsp-pref-sig-algs  }

   id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

   PreferredSignatureAlgorithms ::= SEQUENCE OF
                                      PreferredSignatureAlgorithm

   PreferredSignatureAlgorithm ::= SEQUENCE {
    sigIdentifier       AlgorithmIdentifier{SIGNATURE-ALGORITHM, {...}},
    pubKeyAlgIdentifier SMIMECapability{PUBLIC-KEY, {...}} OPTIONAL  }

 END
```

A.2 1988 ASN.1 Module

```
 OCSP-AGILITY-88 { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-ocsp-agility-2009-88(67) }

 DEFINITIONS EXPLICIT TAGS ::=
 BEGIN

   -- EXPORTS ALL;
   IMPORTS

   id-pkix-ocsp  -- From [RFC2560]
     FROM OCSP
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}

   AlgorithmIdentifier
     FROM PKIX1Explicit88 -- From [RFC5280]
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) };

   SMIMECapability
     FROM SecureMimeMessageV3dot1 -- From [RFC5751]
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) msg-v3dot1(21) }

     id-pkix-ocsp-pref-sig-algs OBJECT IDENTIFIER ::= { id-pkix-ocsp 8 }

     PreferredSignatureAlgorithms ::= SEQUENCE OF
                                      PreferredSignatureAlgorithm

     PreferredSignatureAlgorithm ::= SEQUENCE {
        sigIdentifier        AlgorithmIdentifier,
        pubKeyAlgIdentifier  SMIMECapability OPTIONAL
        }

 END
```

Author's Address

Phillip Hallam-Baker
Default Deny Security

Email: phill@hallambaker.com

Stefan Santesson
3xA Security AB
Sweden

Email: sts@aaa-sec.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 15, 2011

J. Schaad
Soaring Hawk Consulting

December 12, 2010

S/MIME Capabilities for Public Key Definitions
draft-ietf-pkix-pubkey-caps-01

Abstract

This document defines a set of S/MIME Capability types for ASN.1 encoding for the current set of public keys define in the PKIX working group.

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 http://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 June 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

In the process of dealing with the OCSP agility issues in [I-D.ietf-pkix-ocspagility] it was noted that we really wanted to describe information to be used in selecting a public key, but we did not currently have any way of doing so at the current time. This document fills that hole by defining a set of S/MIME Capability types for a small set of public key representations.

S/MIME Capabilities where originally defined in [SMIMEv3-MSG] as a way for the sender of an S/MIME message to tell the recipient of the message the set of encryption algorithms that were supported by the senders system. In the beginning, the focus was primarily on communicating the set of encryption algorithms that were supported. Over time it was expanded to allow for an S/MIME client to say that it supported the compression data type and binary contents. As originally defined it was targeted towards supporting items with a small number of possible parameters. For the RC2 encryption algorithm only two values from the entire range of values were ever use. The object of restricting the set of values was so that a client could do a simple binary comparison without having to decode the S/MIME capability. This was especially easy since most just consisted of the object identifier for the algorithm.

Given that we are assigning different data types to the algorithm descriptors here, and many of the algorithm descriptors are the same as are used in signature, key transport or key agreement algorithms, the public key versions of these structures MUST NOT be placed in the same locations as the other versions. It is expected that the places where one needs S/MIME capabilities for public keys is going to be vastly different than for the other values.

1.1. ASN.1 Notation

The main body of the text is written using snippets of ASN.1 that are extracted from the ASN.1 2008 module in Appendix A. This is because I am a strong advocate of moving to the current versions of ASN.1 as they can contain meta-data which is not representable in the 1988 version of ASN.1. In keeping with the current policy of the PKIX working group, the 1988 module is still to be considered the normative module in the event of a conflict between the contents of the two modules.

When reading this document, it is assumed that you will have a degree of familiarity with the basic object module that is presented in section 3 of RFC 5912 ([RFC5912]). We use the SMIME-CAPS object in this document, it associates two fields together in a single object.

```
SMIME-CAPS ::= CLASS {
    &id         OBJECT IDENTIFIER UNIQUE,
    &Type       OPTIONAL
}
WITH SYNTAX { [TYPE &Type] IDENTIFIED BY &id }
```

These fields are:

&id contains an object identifier. When placed in an object set, this element is tagged so that no two elements can be placed in the set that have the same value in the &id field. Note that this is not a restriction which says that only a single object can exist with a single object identifier.

&Type optionally contains an ASN.1 type identifier. If the field &Type is not defined then the optional parameters field of the AlgorithmIdentifier type would be omitted.

The class also has a specialized syntax for how to define an object in this class. The all upper case words TYPE IDENTIFIER and BY are syntactic sugar to make it easier to read. The square brackets defined optional pieces of the syntax.

One of the things that can be done is to reference the fields of an object while defining other objects. This means that if an object called foo has a field named &value, the value can be directly referenced as foo.&value. This means that we automatically get any updates to values or types and we do not need to do any replication of the data.

1.2. Requirements Terminology

When capitalized the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. RSA Public Keys

There are currently three different public key object identifiers for RSA public keys. These are RSA, RSA ES-OCSP and RSA SSA-PSS.

2.1. Generic RSA Public Keys

Almost all RSA keys that are contained in certificates today use the generic RSA public key format and identifier. This allows for the public key to be used both for key transport and for signature validation (assuming it is compatible with the bits in the key usage extension). The only reason for using one of more specific public key identifiers is if the user wants to restrict the usage of the RSA public key with a specific algorithm.

For the generic RSA public key, the S/MIME capability that is advertised is a request for a specific key size to be used. This would normally be used for dealing with a request on the key to be used for a signature that the client would then verify. In general the user would provide a specific key when a key transport algorithm is being considered.

The ASN.1 that is used for the generic RSA public key is defined as below:

```
scap-pk-rsa SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsa.&id
}

RSAKeyCapabilities ::= SEQUENCE {
   minKeySize        RSAKeySize,
   maxKeySize        RSAKeySize OPTIONAL
}

RSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 |
                        4096 | 8192, ...)
```

In the above ASN.1 we have defined the following:

scap-pk-rsa is a new SMIME-CAP object. This object associates the existing object identifier (rsaEncryption) used for the public key in certificates (defined in [RFC3279] and [RFC5912]) with a new type defined in this document.

RSAKeyCapabilities carries the set of desired capabilities for an RSA key. The fields of this type are:

minKeySize contains the minimum length of the RSA modulus to be used. This field SHOULD NOT contain a value less than 1024.

maxKeySize contains the maximum length of the RSA modules that should be used. If this field is absent then no maximum length is requested/expected. This value is normally selected so as not to cause the current code to run unacceptably long when processing signatures.

RSAKeySize provides a set of suggested values to be used. The values 1024, 2048, 3072, 7680 and 15360 are from the NIST guide on signature sizes [NIST-SIZES] while the others are common powers of two that would be used. The list is not closed and other values can be used.

2.2. RSASSA-PSS Signature Public Keys

While most of the time one will use the generic RSA public key identifier in a certificate, the RSA SSA-PSS identifier can be used if the owner of the key desires to restrict the usage of the key to just this algorithm. This algorithm does have the ability to place a set of algorithm parameters in the public key info structure, they have not been included in this location s as the same information should be carried in the signature S/MIME capabilities instead.

The ASN.1 that is used for the RSA SSA-PSS public key is defined below:

```
scap-pk-rsaSSA-PSS SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaSSA-PSS.&id
}
```

In the above ASN.1 we have defined the following:

scap-pk-rsaSSA-PSS is a new SMIME-CAP object. This object associates the existing object identifier (id-RSASSA-PSS) used for the public key certificates (defined in [RFC4055] and [RFC5912]) with type RSAKeyCapabilities.

2.3. RSA ES-OAEP Key Transport Public Keys

While most of the time one will use the generic RSA public key identifier in a certificate, the RSA ES-OAEP identifier can be used if the owner of the key desires to restrict the usage of the key to

just this algorithm. This algorithm does have the ability to place a set of algorithm parameters in the public key info structure, they have not been included in this location s as the same information should be carried in the key transport S/MIME capabilities instead.

The ASN.1 that is used for the RSA ES-OAEP public key is defined below:

```
scap-pk-rsaES-OAEP SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaES-OAEP.&id
}
```

In the above ASN.1 we have defined the following:

scap-pk-rsaES-OAEP is a new SMIME-CAP object. This object associates the existing object identifier (id-RSAES-OAEP) used for the public key certificates (defined in [RFC4055] and [RFC5912]) with type RSAKeyCapabilities.

3. Diffie-Hellman Keys

There is current two Diffie-Hellman public key object identifiers. These are DH key agreement and DSA.

3.1. DSA Signature Public Key

This public key type is used for the validation of DSA signatures.

The ASN.1 that is used for DSA keys is defined below:

```
scap-pk-dsa SMIME-CAPS ::= {
  TYPE DSAKeyCapabilities
  IDENTIFIED BY pk-dsa.&id
}

DSAKeyCapabilities ::= CHOICE {
    keySizes         [0] SEQUENCE {
       minKeySize            DSAKeySize,
       maxKeySize            DSAKeySize OPTIONAL
    },
    keyParams        [1] pk-dsa.&Params
}

DSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 )
```

In the above ASN.1 we have defined the following:

scap-pk-dsa is a new SMIME-CAP object. This object associated the existing object identifier (id-dsa) used for the public key in certificates (defined in [RFC3279] and [RFC5912]) with a new type defined here, DSAKeyCapabilities.

DSAKeyCapabilities carries the desired set of capabilities for the DSA key. The fields of this type are:

keySizes is used when only a key size is needed to be specified and not a specific group. It is expected that this would be the most commonly used of the two options. In key sizes the fields are used as follows:

minKeySize contains the minimum length of the DSA modulus to be used.

maxKeySize contains the maximum length of the DSA modules that should be used. If this field is absent then no maximum length is requested/expected.

keyParams contains the exact set of DSA for the key used to sign the message.

NOTE: In the original discussions the option keyParams would not have existed in this structure, and they may not exist in a future version of the structure. The issue is that we really only need to have the key size fields, but there seems to be a mis-match between this structure and that used for ECC where we don't specify anything about key sizes, but do specify the exact group to be used. We should probably have a discussion about rationalizing these together.

3.2. DH Key Agreement Keys

This public key type is used with the Diffie-Hellman key agreement algorithm.

The ASN.1 that is used for DH keys is defined below:

```
scap-pk-dh SMIME-CAPS ::= {
  TYPE INTEGER
  IDENTIFIED BY pk-dh.&id
}
```

In the above ASN.1 we have defined the following:

scap-pk-dh is a new SMIME-CAP object. This object associates the existing object identifier (id-dh) used for the public key algorithm in the certificates (defined in [RFC3279] and [RFC5912]) with a new type defined above, DSAKeyCapabilities.

4. Elliptical Curve Keys

There are currently three Elliptical Curve public key object identifiers. These are EC, EC-DH and EC-MQV

4.1. Generic Elliptical Curve Keys

All most all ECC keys that are contained in certificates today use the generic ECC public key format and identifier. This allows for the public key to be used both for key agreement and for signature validation (assuming the appropriate bits are in the certificate). The only reason for using one of the more specific public key identifier is if the user wants to restrict the usage of the ECC public key with a specific algorithm.

For the generic ECC public key, the S/MIME capability that is advertised is a request for a specific group to be used.

The ASN.1 that is used for the generic ECC public key is defined as below:

```
scap-pk-ec SMIME-CAPS ::= {
   TYPE pk-ec.&Type
   IDENTIFIED BY pk-ec.&id
}
```

In the above ASN.1 we have defined the following:

scap-pk-ec is a new SMIME-CAP object. This object associated the existing object identifier (id-ecPublicKey) used for the public key algorithm in the certificates (defined in [RFC3279] and [RFC5912]) with the same type used for the public key (ECPoint).

4.2. Elliptical Curve DH Keys

This public key type is used with the Elliptical Curve Diffie-Hellman key agreement algorithm.

The ASN.1 that is used for EC-DH keys is defined below:

```
scap-pk-ecDH SMIME-CAPS ::= {
  TYPE pk-ecDH.&Type
  IDENTIFIED BY pk-ecDH.&id
}
```

In the above ASN.1 we have defined the following:

scap-ec-dh is a new SMIME-CAP object. This object associated the existing object identifier (id-??) used for the public key algorithm in the certificate (defined in [RFC3279] and [RFC5912]) with the same type structure used for public keys.

4.3. Elliptical Curve MQV Keys

This public key type is used with the Elliptical Curve MQV key agreement algorithm.

The ASN.1 that is used for EC-MQV keys is defined below:

```
scap-pk-ecMQV SMIME-CAPS ::= {
  TYPE pk-ecMQV.&Type
  IDENTIFIED BY pk-ecMQV.&id
}
```

In the above ASN.1 we have defined the following:

scap-ec-MQV is a new SMIME-CAP object. This object associated the existing object identifier (id-??) used for the public key algorithm in the certificate (defined in [RFC3279] and [RFC5912]) with the same type structure used for public keys.

5. RSASSA-PSS Signature Algorithm Capability

This document defines a new S/MIME Capability for the RSA-SSA-PSS signature algorithm. There already exists one in [RFC4055] where the parameters field is not used.

When the S/MIME group defined a S/MIME Capability for the RSA-SSA-PSS signature algorithm, it was done so in the context of how S/MIME defines and uses S/MIME Capabilities. When placed in an S/MIME message [RFC3851] or in a certificate [RFC4262] it is always placed in a sequence of capabilities. This meant that one can place the identifier for RSA-SSA-PSS in the sequence along with the identifier for MD5, SHA-1 and SHA-256. The assumption was then made that one could compute the matrix of all answers and the publisher would support all elements in the matrix. This has the possibility that the publisher could accidently publish a point in the matrix that is not supported.

In this situation, there is only a single item that is published. This means that we need to publish all of the associated information along with the identifier for the signature algorithm in a single entity. For this reason we now define a new parameter type to be used as the S/MIME capability type which contains a hash identifier and a mask identifier. The ASN.1 used for this is as follows:

```
scap-sa-rsaSSA-PSS SMIME-CAPS ::= {
   TYPE RsaSsa-Pss-sig-caps
   IDENTIFIED BY sa-rsaSSA-PSS.&id
}

RsaSsa-Pss-sig-caps ::= SEQUENCE {
   hashAlg  SMIMECapability{{ HashAlgorithms }},
   maskAlg  SMIMECapability{{ MaskAlgorithmSet }} OPTIONAL,
   trailerField INTEGER DEFAULT 1
}

scap-mf-mgf1 SMIME-CAPS ::= {
   TYPE SMIMECapability{{ HashAlgorithms }}
   IDENTIFIED BY id-mgf1
}

MaskAlgorithmSet SMIME-CAPS ::= {scap-mf-mgf1, ...}
```

In the above ASN.1 we have defined the following:

scap-sa-rsaSSA-PSS is a new SMIME-CAP object. This object associates the existing object identifier (id-RSASSA-PSS) used for the signature algorithm (defined in [RFC4055] and [RFC5912]) with the new type RsaSsa-Pss-sig-caps.

RsaSsa-Pss-sig-caps carries the desired set of capabilities for the RSA SSA-PSS signature algorithm. The fields of this type are:

hashAlg contains the S/MIME capability for the hash algorithm we are declaring we support with the RSA-SSA-PSS signature algorithm.

maskAlg contains the S/MIME capability for the mask algorithm we are declaring we support with the RSA-SSA-PSS signature algorithm.

trailerField specifies which trailer field algorithm is being supported. This MUST be the value 1.

NOTE: In at least one iteration of the design we used a sequence of hash identifiers and a sequence of masking functions and again made the assumption that entire matrix would be supported. This has been removed at this point since the original intent of S/MIME capabilities is that one should be able to do a binary comparison of the DER encoding of the field and determine a specific capability was published. We could return back to using the sequence if we wanted to lose the ability to do a binary compare but needed to shorten the encodings. This does not currently appear to be an issue at this point.

6. Security Considerations

This document provides new fields that can be placed in an S/MIME capabilities sequence. There are number of considerations that need to be taking into account when doing this.

As mentioned above, there are a small number of cases where the same object identifier may be used to identify a public key and an algorithm. This is the case for many years with the OID rsaEncryption where it identifies both a public key and the RSA v1.5 key transport algorithm. This means that when an S/MIME capabilities sequence is defined care needs to be taken to specify the types of algorithms and/or public keys that are to be specified in that sequence. In general, it is expected that algorithms and public keys will be segregated.

The more detailed the information that is communicated, the better the end results are going to be. If you can state you do RSA v1.5, EC-DSA, SHA-1 and SHA-256, then it would imply that all four values are supported. It may be however that EC-DSA with SHA-1 is not supported. Not including the SHA-1 hash algorithm could lead to problems as RSA with SHA-1 could be the only point of intersection, but including it means that a result may be returned that cannot be processed.

The more information passed the better. The more choices that are passed, the better the odds that both parties will be able to agree on a common algorithm.

The less information passed the better. Passing too much information can lead to computational issues in trying to deal with the possibilities. This becomes acute when a negotiation over algorithms is going on between multiple parties (such as sending an encrypted S/MIME message) where the amount of memory and processing time can be greatly expanded if there are a large number of choices for each recipient.

Ordering of preference of algorithms is not always supported by all places where S/MIME capabilities are used. The addition of preference ordering greatly complicates the decisions to be used, especially as it is expected that not all parties will agree on the same ordering.

7. IANA Considerations

This document has no IANA considerations.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

8.2. Informative References

[I-D.ietf-pkix-ocspagility] Hallam-Baker, P. and S. Santesson, "OCSP Algorithm Agility", draft-ietf-pkix-ocspagility-08 (work in progress), March 2010.

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[RFC4262] Santesson, S., "X.509 Certificate Extension for Secure/ Multipurpose Internet Mail Extensions (S/MIME) Capabilities", RFC 4262, December 2005.

[SMIMEv3-MSG] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[NIST-SIZES] Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendation for Key Management -- Part 1: General", NIST Special Publication 800-57, March 2007.

Appendix A. 2008 ASN.1 Module

```
   PUBLIC-KEY-SMIME-CAPIBLITIES
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0) TBD5 }
   DEFINITIONS ::=
   BEGIN
      IMPORTS
      SMIME-CAPS, PUBLIC-KEY, SMIMECapability
      FROM AlgoritrithmInformation-2009
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-algorithmInformation-02(58)}

      pk-rsa, pk-dsa, pk-dh, pk-ec, pk-ecDH, pk-ecMQV
      FROM PKIXAlgs-2009
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-algorithms2008-02(56) }

      pk-rsaSSA-PSS, pk-rsaES-OAEP, sa-rsaSSA-PSS,
      HashAlgorithms, id-mgf1
      FROM PKIX1-PSS-OAEP-Algorithms-2009
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-rsa-pkalgs-02(54)}
      ;

      --
      --  Define a set containing all of the S/MIME capabilties defined
      --  by this document
      --

      SMimeCaps SMIME-CAPS ::= {
         PubKeys-SMimeCaps |
         scap-sa-rsaSSA-PSS
      }

      PubKeys-SMimeCaps SMIME-CAPS ::= {
         scap-pk-rsa | scap-pk-rsaSSA-PSS |
         scap-pk-dsa |
         scap-pk-ec | scap-pk-ecDH
      }

      --
      --  We defined RSA keys from the modules RFC3279 and RFC4055
      --
```

```
scap-pk-rsa SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsa.&id
}

RSAKeyCapabilities ::= SEQUENCE {
   minKeySize        RSAKeySize,
   maxKeySize        RSAKeySize OPTIONAL
}

RSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 |
                        4096 | 8192, ...)


scap-pk-rsaES-OAEP SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaES-OAEP.&id
}

scap-pk-rsaSSA-PSS SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaSSA-PSS.&id
}

scap-sa-rsaSSA-PSS SMIME-CAPS ::= {
   TYPE RsaSsa-Pss-sig-caps
   IDENTIFIED BY sa-rsaSSA-PSS.&id
}

RsaSsa-Pss-sig-caps ::= SEQUENCE {
   hashAlg  SMIMECapability{{ HashAlgorithms }},
   maskAlg  SMIMECapability{{ MaskAlgorithmSet }} OPTIONAL,
   trailerField INTEGER DEFAULT 1
}

scap-mf-mgf1 SMIME-CAPS ::= {
   TYPE SMIMECapability{{ HashAlgorithms }}
   IDENTIFIED BY id-mgf1
}

MaskAlgorithmSet SMIME-CAPS ::= {scap-mf-mgf1, ...}

--
--  we define DH/DSA keys from the module RFC3279
--

scap-pk-dsa SMIME-CAPS ::= {
  TYPE DSAKeyCapabilities
```

```
      IDENTIFIED BY pk-dsa.&id
    }

    DSAKeyCapabilities ::= CHOICE {
        keySizes         [0] SEQUENCE {
           minKeySize            DSAKeySize,
           maxKeySize            DSAKeySize OPTIONAL
        },
        keyParams        [1] pk-dsa.&Params
    }

    DSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 )

    scap-pk-dh SMIME-CAPS ::= {
      TYPE INTEGER
      IDENTIFIED BY pk-dh.&id
    }

    --
    --  we define Eliptical Curve keys from the module RFC3279
    --

    scap-pk-ec SMIME-CAPS ::= {
       TYPE pk-ec.&Type
       IDENTIFIED BY pk-ec.&id
    }

    scap-pk-ecDH SMIME-CAPS ::= {
      TYPE pk-ecDH.&Type
      IDENTIFIED BY pk-ecDH.&id
    }

    scap-pk-ecMQV SMIME-CAPS ::= {
      TYPE pk-ecMQV.&Type
      IDENTIFIED BY pk-ecMQV.&id
    }

  END
```

Appendix B. Future Work

A future revision of [RFC5912] should be done at some point which expands the definition of the PUBLIC-KEY class and allows for an S/MIME Capability to be included in the class definition. This would encourage people to think about this as an issue when defining new public key structures in the future.

Author's Address

Jim Schaad
Soaring Hawk Consulting

Email: jimsch@augustcellars.com

Network Working Group
Internet-Draft
Updates: 5272, 5273, 5274
(if approved)
Intended status: Standards Track
Expires: July 15, 2011

J. Schaad
Soaring Hawk Consulting

January 11, 2011

Certificate Management over CMS (CMC) Updates
draft-ietf-pkix-rfc5272-bis-02

Abstract

This document contains a set of updates to the base syntax for CMC, a Certificate Management protocol using the Cryptographic Message Syntax (CMS). This document updates RFC 5272, RFC 5273 and RFC 5274.

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 http://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 July 15, 2011.

Table of Contents

1. Introduction

While dealing with the Suite B profile of CMC [I-D.turner-suiteb-cmc], a number of defencies where noted in the current base CMC specification. This document has a set of updates to [RFC5272], [RFC5273] and [RFC5274] to deal with those issues.

1.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Updates to RFC 5272 - Certificate Management over CMS (CMC)

2.1. New Section 1.3. Changes Since RFC 5272

This section is inserted before the current section 1.3.

The following changes were made in this document.

Addition of new controls:

RA Identity Witness allows for an RA to perform the identity checking using the identity and shared-secret and then tell any following servers that the identity check was successfully performed.

Change Subject Name allows for a client to request a change in the subject name and subject alternate name fields in a certificate.

Response Body allows for an RA to identify a nested response for an EE to process.

Add Extended Key Usages for CMC - Define a new Subject Information Access to hold locations to contact the CMC server at.

2.2. Replace Section 6.3. Linking Identity and POP Information

In a Full PKI Request, identity information about the client is carried in the signature of the SignedData containing all of the certification requests. Proof-of-possession information for key pairs, however, is carried separately for each PKCS #10 or CRMF certification request. (For keys capable of generating a digital signature, the POP is provided by the signature on the PKCS #10 or CRMF request. For encryption-only keys, the controls described in Section 6.7 are used.) In order to prevent substitution-style attacks, the protocol must guarantee that the same entity generated both the POP and proof-of-identity information.

This section describes three mechanisms for linking identity and POP information: witness values cryptographically derived from the shared-secret (Section 6.3.1), shared-secret/subject name matching (Section 6.3.2), and linking with an existing certificate (6.3.3). Clients and servers MUST support the witness value and certificate linking techniques. Clients and servers MAY support shared-secret/ name matching or other bilateral techniques of similar strength. The idea behind the first two mechanisms is to force the client to sign some data into each certification request that can be directly associated with the shared-secret; this will defeat attempts to

include certification requests from different entities in a single Full PKI Request.

2.3. Replace Section 6.3.3. Renewal and Rekey Messages

New section title is "Existing Certificate Linking"

Linking between the POP and an identity is easy when an existing certificate is used. The client copies all of the naming information from the existing certificate (name and subject alternative name) into the new certification request. The POP on the certificate is then performed by using the new key to sign the identity information. The identity information is then tied back by signing the POP proof as part of the PKIData with a certificate that has matching identity information.

Existing certificate linking can be used in the following circumstances:

When replacing a certificate by doing a renewal or rekey certification request.

Use of an existing certificate to get a new certificate. An example of this would be to get a key establishment certificate after having gotten a signature certificate.

Use of a third party certificate to get a new certificate from a CA. An example of this would be to use a certificate and key pair distributed with a device to prove an identity. This would require that the CA have an out-of-band channel to map the device identity to the EE identity.

2.4. New Section 6.20 RA Identity Proof Witness control

Editors Note: The control is to be added to the table of controls in section 6.

The RA Identity Proof Witness control allows an RA to indicate to subsequence control processors that all of the identity proof requirements have been met. This permits the identity proof to be performed at a location closer to the end-entity. For example, the identity proof could be done within a department while the CA could be companywide. The RA would perform the identity proof, and potentially other tasks that require the secret to be used, while the CA would be prevented from knowing the same information. If the identity proof fails, then the RA returns an error to the client noting that fact.

The relevant ASN.1 for the RA Identity Proof Witness control is as follows:

```
cmc-raIdentityWitness CMC-CONTROL ::=
   { BodyPartPath IDENTIFIED BY id-cmc-raIdentityWitness }

id-cmc-raIdentityWitness OBJECT IDENTIFIER ::= {id-cmc TBD}
```

The object defined for this control is a CMC-CONTROL and is named cmc-raIdentityWitness. The object is added to the object set Cmc-Control-Set. The control is only permitted to occur in the control sequence of a PKIData object. It is not permitted to occur in the control sequence of a PKIRespones. The control is only permitted to be used by an RA. The control may occur multiple times in a control sequence.

The control is identified using the object identifier id-cmc-raIdentityWitness.

The type structure associated with the control is BodyPartPath. The path contains a sequence of body part identifiers one of the following items:

Identity Proof control if the RA verified the identity proof in this control.

Identity Proof Version 2 if the RA verified the identity proof in this control.

Full PKI Request if the RA performed an out-of-band identity proof for this request. The request SHOULD NOT contain either Identity Proof control.

Simple PKI Request if the RA performed an out-of-band identity proof for this request.

The RA Identity Proof Witness control will frequently be associated with a Modify Certification Request control which changes the name fields in the associated certification requests as the RA will frequently know the actual name to be assigned to the entity requesting the certificate and the entity will not know the actual details of the name. (The association would be setup by the operator at the time the shared secret was generated by the RA.)

When this control is placed in a message, it is RECOMMENDED that the Control Processed Control be placed in the body sequence as well. Using the explicit new control, rather than implicitly relying on the Control Processed control is important due to the need to explicity

know which identity proofs have been perform. The new control also allows an RA to state that out-of-band idenitity proofs have been performed.

When the identity proof is performed by an RA, the RA also needs to perform the linking between the identity proof and the name information wrapped inside of the key proof-of-possession.

2.5. New Section 6.21 Change Subject Name Control

This item is to be added to the table in section 6.

The Client Name Change Request Control is designed for a client to ask for a change in its name as part of a certificate. This cannot be done in the simple way of just changing the requested subject name in the certificate template because of security issues. The name in the certificate request needs to match the name in the certificate used to sign the request in order that identity and possession proofs are correctly applied.

The relevant ASN.1 for the Client Name Change Request control is as follows:

```
at-cmc-changeSubjectName ATTRIBUTE ::=
   { ChangeSubjectName IDENTIFIED BY id-cmc-changeSubjectName }

id-cmc-changeSubjectName OBJECT IDENTIFIER ::= {id-cmc TBD}

ChangeSubjectName ::= SEQUENCE {
    subject             Name OPTIONAL,
    subjectAlt          SubjectAltName OPTIONAL
}
(WITH COMPONENTS {..., subject PRESENT} |
      COMPONENTS {..., subjectAltPRESENT} )
```

The control is designed to be used as an ATTRIBUTE object. As such the control is placed in one of the following two places:

The attributes field in a CertificationRequest.

The controls field of a CertRequest for a CRMF certification request.

The control is identified by the Object Identifier id-cmc-changedSubjectName.

The ASN.1 type associated with control is ChangeSubjectName. The fields of the structure are configured as follows:

subject contains the requested subject name for the new certificate.

subjectAlt contains the requested subject alternative name for the new certificate.

At least one of the fields in the sequence MUST be present when encoding the structure.

When the CA processes this attribute in a certification request it will do the following:

1. The subject field is copied to the name field of the template if present. If the subject field is absent, the name field of the template will be set to a empty sequence.

2. The subjectAlt field is used as the content of a SubjectAltName extension in the certificate if present. The subjectAltName extension is removed from the certificate template if the subjectAlt field is absent.

2.6. New Section 6.22 Response Body Control

This item is to be added to the table in section 6.

The Response Body Control is designed for an RA to inform an EE that there is an embedded response message that needs to be processed as part of the processing of this message. This control is designed to be used in a couple of different cases where an RA has done some additional processing on the certificate request such as key generation and needs to respond with both the original response message from the certificate issues as well as the response the RA is generating. Another case this is useful is the shared secret is between the RA and the EE and the RA needs to return a Publish Trust Anchors control in order to populate the correct trust points.

The relaveant ASN.1 for the Response Body Control is as follows:

```
cmc-responseBody CMC-CONTROL ::= {
   BodyPartPath IDENTIFIED BY id-cmc-responseBody
}

id-cmc-responseBody OBJECT IDENTIFIER ::= {id-cmc TBD}
```

This object id defined as a CMC control and is named cmc-responseBody. The control is added to the Cmc-Control-Set. The control is only permitted to occur in the control sequence of a PKIResponse object. It is not permitted to occur in the control sequence of a PKIData. The control is only permitted to be used by

an RA (a CA has no need to generate nested responses targetted at the EE). The control may occur multiple times in a single control sequence.

The control is identified using the object identifier id-cmc-responseBody.

The type structure associated with the control is BodyPartID. This field controls the control number for the nested response.

2.7. New Section 8. Certificate Requirements

This section is to be inserted before the current section 8.

Certificates for servers used in the CMC protocol SHOULD conform with the profile defined in [RFC5280]. This document defines some additional items that can appear in CMC server certificates. Section 8.1 defines some additional Extended Key Usage values that can appear in certificates. Section 8.2 defines a new Subject Information Access value which allows for a CMC certificate to publish information on how to contact the services it provides.

2.8. New Section 8.1. Extended Key Usage

The Extended Key Usage (EKU) extension is used to restrict the use of a certificate to specific applications. We define three different EKUs in this document. The ASN.1 to define these EKUs is:

```
id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp TBD }
id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp TBD }
id-kp-cmcArchive OBJECT IDENTIFIER ::= { id-kp TBD }
```

The usage description for each of the EKUs is as follows:

CMC Certification Authorities are identified by the id-kp-cmcCA extended key usage. The certificate may be the same as the CA certificate or may be different than the CA certificate. If a different certificate is used, the certificates containing the id-kp-cmcCA extended key usage SHOULD have the same name as the certificate used for issuing the certificates. (Using a separate public key for CMC protocol operations and for issuing Certificates and CRLs decreases the number of operations for which the private key would be used.)

CMC Registration Authorities are identified by the id-kp-cmcRA extended key usage. This usage is placed into RA certificates.

CMC Archive Servers are identified by the id-kp-cmcArchive extended key usage. CMC Archive Servers and the associated protocol are to be defined in a future document.

2.9. New Section 8.2. Subject Information Access

The subject information access extension indicates how to access the information and services for the subject of the certificate. We define a new value to go into this extension to identify the different locations that CMC services will be available. If this value is placed in a certificate, an appropriate extended key usage defined in section 8.1 MUST be included in the certificate as well.

The id-ad-cmc OID is used when the subject offers certification services using the CMC Protocol. Where the CMC services are available via HTTP or FTP, accessLocation MUST be a uniformResourceIdentifier. Where the CMC services are available via electronic mail, accessLocation MUST be an rfc822Name. Where CMC services are available using TCP/IP, the dNSName or iPAddress name forms MUST be used. Since the GeneralName data structure does not permit the inclusion of a port number, in the absence of other external configuration information the value of TBD1 should be used. The semantics of other name forms of accessLocation (when accessMethod is id-ad-cmc) are not defined by this specification.

The ASN.1 for this extension is: GeneralName

```
id-ad-cmc OBJECT IDENTIFIER ::= { id-ad TBD }
```

3. Updates to RFC 5273 - Certificate Management over CMS (CMC): Transport Protocols

3.1. Update to Section 5 TCP-Based Protocol

The following replaces paragraph 3 in section 5.

CMC requires a registered port number [IANA] to send and recieve CMC messages over TCP. The title of this IP Protocol number is "pkix-cmc". The value of this TCP port is TBD.

3.2. New Section IANA Considerations

This is a new section to be inserted before the current section 6.

Keyword	Decimal	Description	References
pkix-cmc	TBD1/tcp	PKIX CMC	[TBA]

4. Updates to RFC 5274 - Certificate Management Message over CMS (CMC): Compliance Requirements

4.1. Update to Section 4.2 Controls

The following lines should be added to the end of Table 1.

RaIdentityWitness N/A MUST (2)

ChangeSubjectName MAY N/A MUST

5. IANA Considerations

This document contains a new IANA considerations section to be added to [RFC5273] as part of this update.

6. Security Considerations

To be supplied.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.

[RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, June 2008.

[RFC5274] Schaad, J. and M. Myers, "Certificate Management Messages over CMS (CMC): Compliance Requirements", RFC 5274, June 2008.

[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, May 2008.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

7.2. Informational References

[I-D.turner-suiteb-cmc] Zieglar, L., Peck, M., and S. Turner, "Suite B Profile of Certificate Management over CMS", draft-turner-suiteb-cmc-03 (work in progress), June 2010.

Appendix A. ASN.1 Module

An updated 2009 ASN.1 module has been provided as part of this update. The module contains changes that were made as part of the re-write to current ASN.1 standards in [RFC5912] as well as the changes for this document.

```
EnrollmentMessageSyntax-2009
    {iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechansims(5) pkix(7) id-mod(0)
    id-mod-cmc2002-02(TBD49)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
EXPORTS ALL;
IMPORTS

AttributeSet{}, Extension{}, EXTENSION, ATTRIBUTE
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)}

AlgorithmIdentifier{}, DIGEST-ALGORITHM, KEY-WRAP, KEY-DERIVATION,
    MAC-ALGORITHM, SIGNATURE-ALGORITHM, PUBLIC-KEY
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)}

CertificateSerialNumber, GeneralName, CRLReason, ReasonFlags,
    CertExtensions, GeneralNames
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)}

Name, id-pkix, PublicKeyAlgorithms, SignatureAlgorithms, id-ad, id-kp
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)}

ContentInfo, IssuerAndSerialNumber, CONTENT-TYPE
FROM CryptographicMessageSyntax-2009
  { iso(1) member-body(2) us(840) rsadsi(113549)
     pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

CertReqMsg, PKIPublicationInfo, CertTemplate
FROM PKIXCRMF-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005-02(55)}
```

```
  mda-sha1
  FROM PKIXAlgs-2009
       { iso(1) identified-organization(3) dod(6)
       internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-pkix1-algorithms2008-02(56)}

  kda-PBKDF2, maca-hMAC-SHA1
  FROM CryptographicMessageSyntaxAlgorithms-2009
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cmsalg-2001-02(37) }

  mda-sha256
  FROM PKIX1-PSS-OAEP-Algorithms-2009
       { iso(1) identified-organization(3) dod(6)
         internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkix1-rsa-pkalgs-02(54) } ;

  --  CMS Content types defined in this document

  CMC-ContentTypes CONTENT-TYPE ::= { ct-PKIData | ct-PKIResponse, ... }

  --  Signature Algorithms defined in this document

  SignatureAlgs SIGNATURE-ALGORITHM ::= { sa-noSignature }

  --  CMS Unsigned Attributes

  CMC-UnsignedAtts ATTRIBUTE ::= { aa-cmc-unsignedData }

  --
  --

  id-cmc OBJECT IDENTIFIER ::= {id-pkix 7}   -- CMC controls
  id-cct OBJECT IDENTIFIER ::= {id-pkix 12}  -- CMC content types

  -- This is the content type for a request message in the protocol

  ct-PKIData CONTENT-TYPE ::=
      { TYPE PKIData IDENTIFIED BY id-cct-PKIData }
  id-cct-PKIData OBJECT IDENTIFIER ::= { id-cct 2 }

  PKIData ::= SEQUENCE {
      controlSequence    SEQUENCE SIZE(0..MAX) OF TaggedAttribute,
      reqSequence        SEQUENCE SIZE(0..MAX) OF TaggedRequest,
      cmsSequence        SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,
      otherMsgSequence   SEQUENCE SIZE(0..MAX) OF OtherMsg
  }
```

```
  BodyPartID ::= INTEGER(0..4294967295)

  TaggedAttribute ::= SEQUENCE {
      bodyPartID         BodyPartID,
      attrType           CMC-CONTROL.&id({Cmc-Control-Set}),
      attrValues         SET OF CMC-CONTROL.
                             &Type({Cmc-Control-Set}{@attrType})
  }

  Cmc-Control-Set CMC-CONTROL ::= {
      cmc-identityProof | cmc-dataReturn | cmc-regInfo |
      cmc-responseInfo | cmc-queryPending | cmc-popLinkRandom |
      cmc-popLinkWitness | cmc-identification | cmc-transactionId |
      cmc-senderNonce | cmc-recipientNonce | cmc-statusInfo |
      cmc-addExtensions | cmc-encryptedPOP | cmc-decryptedPOP |
      cmc-lraPOPWitness | cmc-getCert | cmc-getCRL |
      cmc-revokeRequest | cmc-confirmCertAcceptance |
      cmc-statusInfoV2 | cmc-trustedAnchors | cmc-authData |
      cmc-batchRequests | cmc-batchResponses | cmc-publishCert |
      cmc-modCertTemplate | cmc-controlProcessed |
      cmc-identityProofV2 | cmc-popLinkWitnessV2, ...,
      cmc-raIdentityWitness }

  OTHER-REQUEST ::= TYPE-IDENTIFIER

  --  We do not define any other requests in this document
  --     examples might be attribute certification requests

  OtherRequests OTHER-REQUEST ::= {...}

  TaggedRequest ::= CHOICE {
      tcr               [0] TaggedCertificationRequest,
      crm               [1] CertReqMsg,
      orm               [2] SEQUENCE {
          bodyPartID            BodyPartID,
          requestMessageType    OTHER-REQUEST.&id({OtherRequests}),
          requestMessageValue   OTHER-REQUEST.&Type({OtherRequests}
                                    {@.requestMessageType})
      }
  }

  TaggedCertificationRequest ::= SEQUENCE {
      bodyPartID           BodyPartID,
      certificationRequest  CertificationRequest
  }

  AttributeList ATTRIBUTE ::= {at-extension-req, ...,
      at-cmc-changeSubjectName}
```

```
  CertificationRequest ::= SEQUENCE {
     certificationRequestInfo  SEQUENCE {
         version                   INTEGER,
         subject                   Name,
         subjectPublicKeyInfo      SEQUENCE {
             algorithm                 AlgorithmIdentifier{PUBLIC-KEY,
                                           {PublicKeyAlgorithms}},
             subjectPublicKey          BIT STRING
         },
         attributes                [0] IMPLICIT SET OF
                                       AttributeSet{{AttributeList}}
     },
     signatureAlgorithm        AlgorithmIdentifier
                                   {SIGNATURE-ALGORITHM,
                                       {SignatureAlgorithms}},
     signature                 BIT STRING
  }

  TaggedContentInfo ::= SEQUENCE {
      bodyPartID              BodyPartID,
      contentInfo             ContentInfo
  }

  OTHER-MSG ::= TYPE-IDENTIFIER

  --  No other messages currently defined

  OtherMsgSet OTHER-MSG ::= {...}

  OtherMsg ::= SEQUENCE {
      bodyPartID        BodyPartID,
      otherMsgType      OTHER-MSG.&id({OtherMsgSet}),
      otherMsgValue     OTHER-MSG.&Type({OtherMsgSet}{@otherMsgType}) }

  --  This defines the response message in the protocol

  ct-PKIResponse CONTENT-TYPE ::=
      { TYPE PKIResponse IDENTIFIED BY id-cct-PKIResponse }
  id-cct-PKIResponse OBJECT IDENTIFIER ::= { id-cct 3 }

  ResponseBody ::= PKIResponse

  PKIResponse ::= SEQUENCE {
      controlSequence   SEQUENCE SIZE(0..MAX) OF TaggedAttribute,
      cmsSequence       SEQUENCE SIZE(0..MAX) OF TaggedContentInfo,
      otherMsgSequence  SEQUENCE SIZE(0..MAX) OF OtherMsg
  }
```

```
  CMC-CONTROL ::= TYPE-IDENTIFIER

  -- The following controls have the type OCTET STRING

  cmc-identityProof CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-identityProof }
  id-cmc-identityProof OBJECT IDENTIFIER ::= {id-cmc 3}

  cmc-dataReturn CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-dataReturn }
  id-cmc-dataReturn OBJECT IDENTIFIER ::= {id-cmc 4}

  cmc-regInfo CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-regInfo }
  id-cmc-regInfo OBJECT IDENTIFIER ::= {id-cmc 18}

  cmc-responseInfo CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-responseInfo }
  id-cmc-responseInfo OBJECT IDENTIFIER ::= {id-cmc 19}

  cmc-queryPending CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-queryPending }
  id-cmc-queryPending OBJECT IDENTIFIER ::= {id-cmc 21}

  cmc-popLinkRandom CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-popLinkRandom }
  id-cmc-popLinkRandom OBJECT IDENTIFIER ::= {id-cmc 22}

  cmc-popLinkWitness CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-popLinkWitness }
  id-cmc-popLinkWitness OBJECT IDENTIFIER ::= {id-cmc 23}

  -- The following controls have the type UTF8String

  cmc-identification CMC-CONTROL ::=
      { UTF8String IDENTIFIED BY id-cmc-identification }
  id-cmc-identification OBJECT IDENTIFIER ::= {id-cmc 2}

  -- The following controls have the type INTEGER

  cmc-transactionId CMC-CONTROL ::=
      { INTEGER IDENTIFIED BY id-cmc-transactionId }
  id-cmc-transactionId OBJECT IDENTIFIER ::= {id-cmc 5}

  -- The following controls have the type OCTET STRING

  cmc-senderNonce CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-senderNonce }
```

```
  id-cmc-senderNonce OBJECT IDENTIFIER ::= {id-cmc 6}

  cmc-recipientNonce CMC-CONTROL ::=
      { OCTET STRING IDENTIFIED BY id-cmc-recipientNonce }
  id-cmc-recipientNonce OBJECT IDENTIFIER ::= {id-cmc 7}

  -- Used to return status in a response

  cmc-statusInfo CMC-CONTROL ::=
      { CMCStatusInfo IDENTIFIED BY id-cmc-statusInfo }
  id-cmc-statusInfo OBJECT IDENTIFIER ::= {id-cmc 1}

  CMCStatusInfo ::= SEQUENCE {
      cMCStatus       CMCStatus,
      bodyList        SEQUENCE SIZE (1..MAX) OF BodyPartID,
      statusString    UTF8String OPTIONAL,
      otherInfo       CHOICE {
         failInfo         CMCFailInfo,
         pendInfo         PendInfo
      } OPTIONAL
  }

  PendInfo ::= SEQUENCE {
      pendToken        OCTET STRING,
      pendTime         GeneralizedTime
  }

  CMCStatus ::= INTEGER {
      success         (0),
      failed          (2),
      pending         (3),
      noSupport       (4),
      confirmRequired (5),
      popRequired     (6),
      partial         (7)
  }

  CMCFailInfo ::= INTEGER {
      badAlg          (0),
      badMessageCheck (1),
      badRequest      (2),
      badTime         (3),
      badCertId       (4),
      unsuportedExt   (5),
      mustArchiveKeys (6),
      badIdentity     (7),
      popRequired     (8),
      popFailed       (9),
```

```
      noKeyReuse      (10),
      internalCAError (11),
      tryLater        (12),
      authDataFail    (13)
  }

  -- Used for RAs to add extensions to certification requests

  cmc-addExtensions CMC-CONTROL ::=
      { AddExtensions IDENTIFIED BY id-cmc-addExtensions }
  id-cmc-addExtensions OBJECT IDENTIFIER ::= {id-cmc 8}

  AddExtensions ::= SEQUENCE {
      pkiDataReference    BodyPartID,
      certReferences      SEQUENCE OF BodyPartID,
      extensions          SEQUENCE OF Extension{{CertExtensions}}
  }

  cmc-encryptedPOP CMC-CONTROL ::=
      { EncryptedPOP IDENTIFIED BY id-cmc-encryptedPOP }
  cmc-decryptedPOP CMC-CONTROL ::=
      { DecryptedPOP IDENTIFIED BY id-cmc-decryptedPOP }
  id-cmc-encryptedPOP OBJECT IDENTIFIER ::= {id-cmc 9}
  id-cmc-decryptedPOP OBJECT IDENTIFIER ::= {id-cmc 10}

  EncryptedPOP ::= SEQUENCE {
      request       TaggedRequest,
      cms             ContentInfo,
      thePOPAlgID     AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
      witnessAlgID    AlgorithmIdentifier{DIGEST-ALGORITHM,
                          {WitnessAlgs}},
      witness         OCTET STRING
  }

  POPAlgs MAC-ALGORITHM ::= {maca-hMAC-SHA1, ...}
  WitnessAlgs DIGEST-ALGORITHM ::= {mda-sha1, ...}

  DecryptedPOP ::= SEQUENCE {
      bodyPartID      BodyPartID,
      thePOPAlgID     AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
      thePOP          OCTET STRING
  }

  cmc-lraPOPWitness CMC-CONTROL ::=
      { LraPopWitness IDENTIFIED BY id-cmc-lraPOPWitness }

  id-cmc-lraPOPWitness OBJECT IDENTIFIER ::= {id-cmc 11}
```

```
  LraPopWitness ::= SEQUENCE {
      pkiDataBodyid   BodyPartID,
      bodyIds         SEQUENCE OF BodyPartID
  }

  --

  cmc-getCert CMC-CONTROL ::=
      { GetCert IDENTIFIED BY id-cmc-getCert }
  id-cmc-getCert OBJECT IDENTIFIER ::= {id-cmc 15}

  GetCert ::= SEQUENCE {
      issuerName      GeneralName,
      serialNumber    INTEGER }


  cmc-getCRL CMC-CONTROL ::=
      { GetCRL IDENTIFIED BY id-cmc-getCRL }
  id-cmc-getCRL OBJECT IDENTIFIER ::= {id-cmc 16}

  GetCRL ::= SEQUENCE {
      issuerName    Name,
      cRLName       GeneralName OPTIONAL,
      time          GeneralizedTime OPTIONAL,
      reasons       ReasonFlags OPTIONAL }

  cmc-revokeRequest CMC-CONTROL ::=
      { RevokeRequest IDENTIFIED BY id-cmc-revokeRequest}
  id-cmc-revokeRequest OBJECT IDENTIFIER ::= {id-cmc 17}

  RevokeRequest ::= SEQUENCE {
      issuerName            Name,
      serialNumber          INTEGER,
      reason                CRLReason,
      invalidityDate         GeneralizedTime OPTIONAL,
      passphrase            OCTET STRING OPTIONAL,
      comment               UTF8String OPTIONAL }


  cmc-confirmCertAcceptance CMC-CONTROL ::=
      { CMCCertId IDENTIFIED BY id-cmc-confirmCertAcceptance }
  id-cmc-confirmCertAcceptance OBJECT IDENTIFIER ::= {id-cmc 24}

  CMCCertId ::= IssuerAndSerialNumber

  -- The following is used to request V3 extensions be added
  --     to a certificate
```

```
  at-extension-req ATTRIBUTE ::=
      { TYPE ExtensionReq IDENTIFIED BY id-ExtensionReq }
  id-ExtensionReq OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
      rsadsi(113549) pkcs(1) pkcs-9(9) 14}

  ExtensionReq ::= SEQUENCE SIZE (1..MAX) OF
      Extension{{CertExtensions}}

  -- The following allows Diffie-Hellman Certification Request
  --     Messages to be well-formed

  sa-noSignature SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-noSignature
      VALUE NoSignatureValue
      PARAMS TYPE NULL ARE required
      HASHES { mda-sha1 }
  }
  id-alg-noSignature OBJECT IDENTIFIER ::= {id-pkix id-alg(6) 2}

  NoSignatureValue ::= OCTET STRING

  --  Unauthenticated attribute to carry removable data.

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

  aa-cmc-unsignedData ATTRIBUTE ::=
      { TYPE CMCUnsignedData IDENTIFIED BY id-aa-cmc-unsignedData }
  id-aa-cmc-unsignedData OBJECT IDENTIFIER ::= {id-aa 34}

  CMCUnsignedData ::= SEQUENCE {
      bodyPartPath        BodyPartPath,
      identifier          TYPE-IDENTIFIER.&id,
      content             TYPE-IDENTIFIER.&Type
  }

  --  Replaces CMC Status Info
  --

  cmc-statusInfoV2 CMC-CONTROL ::=
      { CMCStatusInfoV2 IDENTIFIED BY id-cmc-statusInfoV2 }
  id-cmc-statusInfoV2 OBJECT IDENTIFIER ::= {id-cmc 25}


  EXTENDED-FAILURE-INFO ::= TYPE-IDENTIFIER

  ExtendedFailures EXTENDED-FAILURE-INFO ::= {...}
```

```
  CMCStatusInfoV2 ::= SEQUENCE {
     cMCStatus             CMCStatus,
     bodyList              SEQUENCE SIZE (1..MAX) OF
                                 BodyPartReference,
     statusString          UTF8String OPTIONAL,
     otherInfo             CHOICE {
         failInfo               CMCFailInfo,
         pendInfo               PendInfo,
         extendedFailInfo       [1] SEQUENCE {
            failInfoOID             TYPE-IDENTIFIER.&id
                                        ({ExtendedFailures}),
            failInfoValue           TYPE-IDENTIFIER.&Type
                                        ({ExtendedFailures}
                                            {@.failInfoOID})
         }
      } OPTIONAL
  }

  BodyPartReference ::= CHOICE {
     bodyPartID           BodyPartID,
     bodyPartPath         BodyPartPath
  }

  BodyPartPath ::= SEQUENCE SIZE (1..MAX) OF BodyPartID

  --  Allow for distribution of trust anchors
  --

  cmc-trustedAnchors CMC-CONTROL ::=
      { PublishTrustAnchors IDENTIFIED BY id-cmc-trustedAnchors }
  id-cmc-trustedAnchors OBJECT IDENTIFIER ::= {id-cmc 26}

  PublishTrustAnchors ::= SEQUENCE {
      seqNumber      INTEGER,
      hashAlgorithm  AlgorithmIdentifier{DIGEST-ALGORITHM,
                         {HashAlgorithms}},
      anchorHashes   SEQUENCE OF OCTET STRING
  }

  HashAlgorithms DIGEST-ALGORITHM ::= {
     mda-sha1 | mda-sha256, ...
  }

  cmc-authData CMC-CONTROL ::=
      { AuthPublish IDENTIFIED BY id-cmc-authData }
  id-cmc-authData OBJECT IDENTIFIER ::= {id-cmc 27}

  AuthPublish ::= BodyPartID
```

```
  --   These two items use BodyPartList

  cmc-batchRequests CMC-CONTROL ::=
      { BodyPartList IDENTIFIED BY id-cmc-batchRequests }
  id-cmc-batchRequests OBJECT IDENTIFIER ::= {id-cmc 28}

  cmc-batchResponses CMC-CONTROL ::=
      { BodyPartList IDENTIFIED BY id-cmc-batchResponses }
  id-cmc-batchResponses OBJECT IDENTIFIER ::= {id-cmc 29}

  BodyPartList ::= SEQUENCE SIZE (1..MAX) OF BodyPartID

  cmc-publishCert CMC-CONTROL ::=
      { CMCPublicationInfo IDENTIFIED BY id-cmc-publishCert }
  id-cmc-publishCert OBJECT IDENTIFIER ::= {id-cmc 30}

  CMCPublicationInfo ::= SEQUENCE {
      hashAlg        AlgorithmIdentifier{DIGEST-ALGORITHM,
                           {HashAlgorithms}},
      certHashes     SEQUENCE OF OCTET STRING,
      pubInfo        PKIPublicationInfo
  }

  cmc-modCertTemplate CMC-CONTROL ::=
      { ModCertTemplate IDENTIFIED BY id-cmc-modCertTemplate }

  id-cmc-modCertTemplate OBJECT IDENTIFIER ::= {id-cmc 31}

  ModCertTemplate ::= SEQUENCE {
      pkiDataReference             BodyPartPath,
      certReferences               BodyPartList,
      replace                      BOOLEAN DEFAULT TRUE,
      certTemplate                 CertTemplate
  }

  -- Inform follow-on servers that one or more controls have
  --     already been processed

  cmc-controlProcessed CMC-CONTROL ::=
      { ControlsProcessed IDENTIFIED BY id-cmc-controlProcessed }
  id-cmc-controlProcessed OBJECT IDENTIFIER ::= {id-cmc 32}

  ControlsProcessed ::= SEQUENCE {
      bodyList              SEQUENCE SIZE(1..MAX) OF BodyPartReference
  }

  --  Identity Proof control w/ algorithm agility
```

```
  cmc-identityProofV2 CMC-CONTROL ::=
      { IdentityProofV2 IDENTIFIED BY id-cmc-identityProofV2 }
  id-cmc-identityProofV2 OBJECT IDENTIFIER ::= { id-cmc 33 }

  IdentityProofV2 ::= SEQUENCE {
      proofAlgID       AlgorithmIdentifier{DIGEST-ALGORITHM,
                           {WitnessAlgs}},
      macAlgId         AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
      witness          OCTET STRING
  }

  cmc-popLinkWitnessV2 CMC-CONTROL ::=
      { PopLinkWitnessV2 IDENTIFIED BY id-cmc-popLinkWitnessV2 }
  id-cmc-popLinkWitnessV2 OBJECT IDENTIFIER ::= { id-cmc 34 }

  PopLinkWitnessV2 ::= SEQUENCE {
      keyGenAlgorithm   AlgorithmIdentifier{KEY-DERIVATION,
                            {KeyDevAlgs}},
      macAlgorithm      AlgorithmIdentifier{MAC-ALGORITHM, {POPAlgs}},
      witness           OCTET STRING
  }

  KeyDevAlgs KEY-DERIVATION ::= {kda-PBKDF2, ...}

  cmc-raIdentityWitness CMC-CONTROL ::=
     { BodyPartPath IDENTIFIED BY id-cmc-raIdentityWitness }

  id-cmc-raIdentityWitness OBJECT IDENTIFIER ::= {id-cmc TBD42}


  --
  --  Allow for an End-Entity to request a change in name
  --  This item is added to RegControlSet in CRMF
  --
  at-cmc-changeSubjectName ATTRIBUTE ::=
     { TYPE ChangeSubjectName IDENTIFIED BY id-cmc-changeSubjectName }

  id-cmc-changeSubjectName OBJECT IDENTIFIER ::= {id-cmc TBD43}

  ChangeSubjectName ::= SEQUENCE {
      subject             Name OPTIONAL,
      subjectAlt          GeneralNames OPTIONAL
  }
  (WITH COMPONENTS {..., subject PRESENT} |
   WITH COMPONENTS {..., subjectAlt PRESENT} )

  --
  --  Key purpose identifiers are in the extended key usage extension
```

```
  --

  id-kp-cmcCA OBJECT IDENTIFIER ::= { id-kp TBD44 }
  id-kp-cmcRA OBJECT IDENTIFIER ::= { id-kp TBD45 }
  id-kp-cmcArchive OBJECT IDENTIFIER ::= { id-kp TBD46 }

  --
  --  Subject Information Access identifier
  --

  id-ad-cmc OBJECT IDENTIFIER ::= { id-ad TBD47 }

  END
```

Author's Address

Jim Schaad
Soaring Hawk Consulting

Email: jimsch@augustcellars.com