Abstract

Current methods of publishing certificates in directory services are restricted to just certificates. This document provides a method of publishing certificates with secondary support information such as the SMimeCapabilities attribute (containing bulk algorithm support) in a way that is both authenticated and bound to a given certificate.

This draft is being discussed on the "ietf-smime" mailing list. To join the list, send a message to <ietf-smime-request@imc.org> with the single word "subscribe" in the body of the message. Also, there is a Web site for the mailing list at <http://www.imc.org/ietf-smime>.

1. Introduction

This document discusses a new method of publishing certificates in a directory to provide authenticated attributes as part of the certificate publishing process. This allows for the addition of information such as the SMimeCapabilities attribute from [SMIME] which contains information about the bulk encryption algorithms supported by the End-Entity’s cryptography module.

Section 2 discusses the current set of publishing methods available for use, along with the benefits and restrictions of each method.
Section 3 covers the definition and properties of a SMimeCertificatePublish object.

Throughout this draft, the terms MUST, MUST NOT, SHOULD, and SHOULD NOT are used in capital letters. This conforms to the definitions in [MUSTSHOULD]. [MUSTSHOULD] defines the use of these key words to help make the intent of standards track documents as clear as possible. The same key words are used in this document to help implementers achieve interoperability.

2. Current Publishing Methods

There are several different ways to publish certificate information. These methods include the userCertificate property in LDAP directories, sending signed objects between users, and transport of certificate files (either bare or as CMS degenerate signed objects). Each of these methods has benefits and drawbacks. Each of these methods will now be briefly discussed.

A public directory may be used to distribute certificates. LDAP currently has the userCertificate property defined just for that purpose. The benefits of using a public directory are that a sender may create an encrypted object for a recipient without first receiving information (such as a signed message) from the recipient. Most public directories currently only contain leaf certificates for individuals in the directory entry for the individual. While some directories, such as X.500 directories, provide for a directory entry to contain the CA certificate, this is not the case for all directories. Outside of the structure of an X.500 directory the problems associated with chaining from the individual’s certificate to the CA’s directory entry in order to obtain it’s certificate is difficult to impossible. This leads to two drawbacks: First, the set of bulk algorithms supported by the recipient is unknown. Second, no additional certificates may be carried which would help in validating the recipient’s certificates.

Using certificate files for certificate distribution has the benefit of already being in wide spread use. (They are commonly used for certificate distribution from Certificate Authorities either as part of the enrollment protocol or from web based repositories.) The degenerate CMS signed object form, certificate files may carry a set of certificates to allow a sender to validate the recipients certificates. However, they suffer from two drawbacks. First, as with the public directory, the additional information is not available as part of the certificate file. Second, the certificate is obtained from either the recipient one is encrypting for or a third party (not a directory).

Using signed objects for certificate distribution has the benefit of allowing additional information such as the SMimeCapabilities attribute to be carried as part of the package. It also allows for the inclusion of additional certificates to be used in verifying the encryption certificate used to build an encrypted object. However, it has the drawback that the initialization process is done via a one-on-one process.
3. SMimeEncryptCerts

When publishing one’s own encryption certificates, it is often advisable to publish a wide selection of certificates to insure maximum interoperability. This section describes an attribute that is used both to identify the set of encryption certificates and to establish the set of bulk encryption algorithms supported by each of the certificates.

The SMimeEncryptCerts attribute is used to identify one’s own encryption certificates to the other party. This attribute is a sequence so that more than one encryption certificate can be identified in a single SignerInfo object. Each certificate is then given a set of capabilities so senders can identify the correct certificate to use for specific capabilities.

The structure and OID for the SMimeEncryptCerts attribute are:

```plaintext
id-aa-smimeEncryptCerts OBJECT IDENTIFIER ::= { iso(1)
                                           member-body(2) us(840)
                                           rsadsi(113549) pkcs(1) pkcs9(9)
                                           smime(16) id-aa(2) 13 }

SMimeEncryptCert ::= SEQUENCE {
    hash           Hash,
    capabilities   SMIMECapabilities
}

SMimeEncryptCerts ::= SEQUENCE OF SmimeEncryptCert

Hash ::= OCTET STRING - SHA1 hash of the certificate
```

When a certificate appears in an SMimeEncryptCerts attribute, the certificate MUST be available to the verifier in a well known location. For plain SignedData objects, this is the certificate bag in the object. (Section 4.5 defines another location for LDAP directories.) The order of certificates in the SMimeEncryptCerts attribute is the preferred order of use by the sender.

If present, the SMimeEncryptCerts attribute MUST be an authenticated attribute; it MUST NOT be an unauthenticated attribute. CMS defines authenticatedAttributes as a SET OF AuthAttribute. A SignerInfo MUST NOT include multiple instances of the SMimeEncryptCerts attribute. CMS defines the ASN.1 syntax for the authenticated attributes to include attrValues SET OF AttributeValue. A SMimeEncryptCerts attribute MUST only include a single instance of AttributeValue. There MUST NOT be zero or multiple instances of AttributeValue present in the attrValues SET OF AttributeValue.

4. SMimeCertificatePublish Object

The structure of the SMimeCertificatePublish object is defined in this section. This object has the benefit that it is published into a directory service (and thus is available to all parties) and it contains a signed object that allows it to carry the additional information desired to increase interoperability.
This section describes the LDAP directory schema, the body content and additional restrictions on the attribute and signers of the SignedData object used in publishing the user’s certificate.

The ASN definition of a SMimeCertificatePublish object is the same a CMS signed object.

```
SMimeCertificatePublish ::= ContentInfo
```

Where the contentType is id-signed-data and the content is a SignedData content.

A SMimeCertificatePublish object MAY contain multiple SignerInfo objects. Each SignerInfo object is independent. This document imposes no restrictions on attributes that appear in more that one SignerInfo object.

4.1 Signed Content

The SMimeCertificatePublish object is explicitly designed to carry no body content. All information is carried in the signed attribute section of the SignerInfo.

The following object identifier is used to distinguish the content of a SMimeCertificatePublish:

```
id-ct-publishCert OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-ct(1) 3}
```

When creating a SMimeCertificatePublish object, the eContent of the Signed-Data object is omitted and the eContentType OID is set to id-ct-publishCert. Note this is different from an empty content, which would be represented as an octet string containing zero bytes. The hash of the body (used in the id-message-digest attribute) is set to the initialization value of the hash function. (This is expected to provide the same result as if you had hashed a body containing exactly 0 bytes.)

4.2 Signed Attributes

The signed attributes section MUST be present in the SignerInfo object, and the following signed attributes MUST be present: The signing-time attribute (from [CMS]), the SMimeCapabilities and SMIMEEncryptionKeyPreference (from [SMIME]).

4.3 CertificateSet

This draft imposes additional restrictions on the set of certificates to be included in the SignedData object beyond those specified in [CMS] and [SMIMECERT]. A chain of certificate from the end-entity certificate(s) to the root certificate(s) MUST be included in the CertificateSet. Unlike in S/MIME messages the root certificate MUST be included in the CertificateSet. The root certificate is included so that end-entities have a better chance of finding and independently verifying the trustworthiness of the root
certificate based on its content.

User agents MUST NOT automatically trust any root certificate found in a SMimeCertificatePublish object.

4.4 Signing Certificate

The SMimeCertificatePublish object MUST be signed by a signing certificate associated with the end-entity, or a signing certificate of a CA in the validation path of the encryption certificate.

Part of the process of extracting certificates involves comparing the certificate found to the address matching the directory look-up. The validation SHOULD match the address used to look up the certificate with one of the names found in the certificate. Thus, if an RFC822 name was used to do the directory look-up, the RFC822 name would be in the SubjectAltName extension on the certificate.

The steps for extracting the encryption certificate from a SMimeCertificatePublish object are as follows:

1. Verify that the SMimeCertificatePublish object contains a valid signature and the certificate used to sign the message can be validated.

2. Does the certificate used to sign the SMimeCertificatePublish object "match" the intended recipient of the encryption object? If so, proceed to step 6 else no encryption certificate is found.

3. Get the set of potential encryption certificates from the SMIMEEncryptCerts attribute in the signed attributes of the SMimeCertificatePublish object.

4. Select the encryption certificate from the set of potential encryption certificates by validating the certificate and examining the set of encryption algorithms.

In all cases, once an encryption certificate has been obtained, the standard methods of validating signatures on the certificate and checking for revocation MUST be followed.

4.5 LDAP Schema

After a SignedData object has been produced, it needs to be published into one or more directories. This section describes the LDAP schema used to support this.

A new LDAP attribute userSMimeCertificate is defined by this document. The attribute is defined according to the syntax provided in [LDAPV3]. The definition of this attribute is:

```
( 1 2 840 113549 1 9 16 <TBD>
 NAME   'userSMimeCertificate'
 SYNTAX 'binary'
 MULTI-VALUE
 USAGE userApplications
)
```
If the SignedData object is to be published in userSMimeCertificate, the end-entity certificates MAY be omitted from the certificate bag and published in the userCertificates LDAP attribute instead.

If the CA is the only entity that can write to the directory, it may wish to provide some mechanism for updating the attributes such as the smimeUserCapabilities in the published object.

4.6 MIME Encoding

The application/pkcs7-mime-publish content type is used to carry SMimeCertificatePublish objects as mime objects. The optional "name" parameter SHOULD be emitted as part of the Content-Type field. The file extension for the file name SHOULD be ".p7p".

A. ASN Module

SMimeCertDistributionSyntax

{ iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-9(9) smime(16) modules(0) <TBD> }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All
-- The types and values defined in this module are exported for
-- use in the other ASN.1 modules. Other applications may use
-- them for their own purposes.

IMPORTS
-- SMime Cryptographic Message Format
  ContentInfo
  FROM CryptographicMessageSyntax { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
    modules(0) cms(1) }

-- SecureMimeMessageV3
  SMIMECapabilities
  FROM SecureMimeMessageV3 { iso(1) member-body(2) us(840)
    rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)
    smime(4));

-- S/MIME Object Identifier Registry
Schaad
CertDist October 1999

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

-- Authenticated Attribute identifying Encryption Certificates
-- Value is a single SMimeEncryptCerts
id-aa-smimeEncryptCerts OBJECT IDENTIFIER ::= { id-smime id-aa(2)
  13 }

SMimeEncryptCerts ::= SEQUENCE OF SMimeEncryptCert

SMimeEncryptCert ::= SEQUENCE {
  hash Hash,
  capabilities SMIMECapabilities
Hash ::= OCTET STRING -- SHA1 hash of the certificate

-- Content Type of Certificate publish message.
-- Signed content is detached and empty
id-ct-publishCert OBJECT IDENTIFIER ::= { id-smime id-ct(1) 3 }

SMimeCertificatePublish ::= ContentInfo

END -- of SMimeCertDistributionSyntax

B. Backwards Compatibility

The SMimeCertificatePublish object is based on work previously done at both Microsoft and Netscape.

Both of these companies have implemented a version of
userSMimeCertificate in their mail LDAP directory structures.
Microsoft has also put the property into its MAPI based directory schema.

Both companies use a ContentInfo object containing a SignedData object with one SignerInfo object. In both cases however the
eContent is tagged with id-data not id-ct-publishCert. The actual content is omitted from the SMimeCertificatePublish object.

In the case of both companies, clients who implement this feature require that the end-entity is the signer of the object; the CA is not permitted to sign and publish the object.

Microsoft has also produced an early version of the
SMimeEncryptCerts attribute. The syntax for this structure is

id-Microsoft-SMimeEncryptCert OBJECT IDENTIFIER ::= {1 3 6 1 4 1 311 16 4}

Microsoft-SMimeEncryptionert ::= IssuerAndSerialNumber

A description of IssuerAndSerialNumber can be find in [CMS].

C. Registration of MIME

To: ietf-types@iana.org
Subject: Registration of MIME media type application/pkcs7-mime-publish

MIME media type name: application
MIME subtype name: pkcs7-mime-publish
Required parameters: none
Optional parameters: name, filename
Encoding considerations: Will be binary data, therefore should use base-64 encoding
Security considerations: There is no requirement for additional
security mechanisms to be applied at this level. The required mechanisms are designed into the SMimeCertificatePublish content.

Interoperability considerations: -

Published specification: this document

Applications that use this media type: Secure Internet mail and other secure data transports.

Additional information:
File extension (s): p7p
Macintosh File Type Code (s): -

Person and email address to contact for further information: Jim Schaad, jimsch@microsoft.com

Intended usage: COMMON

D. Example Message

In this example Alice makes the statement that messages encrypted for her should use one of two encryption certificates issued to Bob.

0 30 NDEF: SEQUENCE {
2 06  9: OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
13 A0 NDEF: [0] {
15 30 NDEF: SEQUENCE {
17 02 1: INTEGER 1
20 31 11: SET {
22 30  9: SEQUENCE {
24 06  5: OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
31 05  0: NULL

33 30 NDEF: SEQUENCE {
35 06 11: OBJECT IDENTIFIER id-ct-publishCert (1 2 840 113549 1 9 16 1 3)
}
}
38 30  5: INTEGER
71 02 16: INTEGER
: 46 34 6B C7 80 00 56 BC 11 D3 6E 2E 9F F2 50 20
89 30 13: SEQUENCE {
91 06  9: OBJECT IDENTIFIER
: sha1withRSAEncryption (1 2 840 113549 1 1 5)
102 05  0: NULL

104 30 18: SEQUENCE {
106 31 16: SET {
108 30 14: SEQUENCE {
110 06  3: OBJECT IDENTIFIER commonName (2 5 4 3)

Schaad
CertDist
October 1999
115 13  7:      PrintableString 'CarlRSA'
        :
        :  
124 30  30:    SEQUENCE {
126 17  13:      UTCTime '990818070000Z'
141 17  13:      UTCTime '991231235959Z'
        :  
156 30  18:    SEQUENCE {
158 31  16:      SET {
160 30  14:        SEQUENCE {
162 06  3:          OBJECT IDENTIFIER commonName (2 5 4 3)
167 13  7:          PrintableString 'CarlRSA'
        :
        :  
176 30  159:    SEQUENCE {
179 30  13:      SEQUENCE {
181 06  9:        OBJECT IDENTIFIER
          :          rsaEncryption (1 2 840 113549 1 1 1)
192 05  0:        NULL
        :  
194 03  141:      BIT STRING 0 unused bits
          :
          :
            30 81 89 02 81 81 00 E4 4B FF 18 B8 24 57 F4 77
            FF 6E 73 7B 93 71 5C BC 33 1A 92 92 72 23 D8 41
            46 D0 CD 11 3A 04 B3 8E AF 82 9D BD 51 1E 17 7A
            :          F2 76 2C 2B B6 39 A7 BD D7 8D 1A 53 EC E4 00 D5
            :          8F B8 4E AB B9 7D D5 96 65 DA 16 A0 C5 BE 0E AE
            :          44 5B EF 5E F4 A7 29 CB 82 DD AC 44 E9 AA 93 94
            :          29 0E F8 18 D6 C8 57 5E F2 76 C4 F2 11 60 38 B9
            :          1B 3C 1D 97 C9 6A F1 02 03 01 00 01
        :  
338 A3  66:    [3] {
340 30  64:      SEQUENCE {
Schaad
CertDist
October 1999

342 30  15:    SEQUENCE {
344 06  3:        OBJECT IDENTIFIER basicConstraints (2 5 29 19)
349 01  1:        BOOLEAN TRUE
352 04  5:        OCTET STRING
              :          30 03 01 01 FF
              :  
359 30  14:    SEQUENCE {
361 06  3:        OBJECT IDENTIFIER keyUsage (2 5 29 15)
366 01  1:        BOOLEAN TRUE
369 04  4:        OCTET STRING
              :          03 02 01 B6
              :  
375 30  29:    SEQUENCE {
377 06  3:        OBJECT IDENTIFIER
              :          subjectKeyIdentifier (2 5 29 14)
382 04  22:        OCTET STRING
              :          04 14 E9 E0 90 27 AC 78 20 7A 9A D3 4C F2 42 37
              :          4E 22 AE 9E 38 BB
              :  
              :  
406 30  13:    SEQUENCE {
408 06 9:  OBJECT IDENTIFIER
    :        sha1withRSAEncryption (1 2 840 113549 1 1 5)
419 05 0:       NULL
    :   }
421 03 129:   BIT STRING 0 unused bits
    :   B7 9E D4 04 D3 ED 29 E4 FF 89 89 15 2E 4C DB 0C
    :   F0 48 0F 32 61 EE C4 04 EC 12 5D 2D FF 0F 64 59
    :   7E 0A C3 ED 18 FD E3 56 40 37 A7 07 B5 F0 38 12
    :   61 50 ED EF DD 3F E3 0B B8 61 A5 A4 9B 3C E6 9E
    :   9C 54 9A B6 95 D6 DA 6C 3B B5 2D 45 35 9D 49 01
    :   76 FA B9 B9 31 F9 F9 6B 12 53 A0 F5 14 60 9B 7D
    :   CA 3E F2 53 6B B0 37 6F AD E6 74 D7 DB FA 5A EA
    :   14 41 63 5D CD BE C8 0E C1 DA 6A 8D 53 34 18 02
553 30 520:   SEQUENCE {
557 30 369:     SEQUENCE {
561 A0 3: [0] {
563 02 1:        INTEGER 2
      
566 02 16:        INTEGER
          :  46 34 6B C7 80 00 56 BC 11 D3 6E 2E CD 5D 71 D0
584 30 13:     SEQUENCE {
586 06 9:        OBJECT IDENTIFIER
          :        sha1withRSAEncryption (1 2 840 113549 1 1 5)
597 05 0:        NULL
      
599 30 18:     SEQUENCE {
601 31 16:      SET {
603 30 14:        SEQUENCE {
605 06 3:         OBJECT IDENTIFIER commonName (2 5 4 3)
610 13 7:           PrintableString ‘CarlRSA’
          :   }
Schaad
CertDist
10
October 1999
      
619 30 30:     SEQUENCE {
621 17 13:      UTCTime ‘990819070000Z’
636 17 13:      UTCTime ‘391231235959Z’
      
651 30 17:     SEQUENCE {
653 31 15:      SET {
655 30 13:        SEQUENCE {
657 06 3:         OBJECT IDENTIFIER commonName (2 5 4 3)
662 13 6:           PrintableString ‘BobRSA’
          :   }
          :   }
670 30 159:     SEQUENCE {
673 30 13:      SEQUENCE {
675 06 9:        OBJECT IDENTIFIER
          :        rsaEncryption (1 2 840 113549 1 1 1)
686 05 0:        NULL
      
688 03 141:   BIT STRING 0 unused bits
    :   30 81 89 02 81 81 00 CA 5C E1 2E EC CF C1 3B 5D
    :   10 1B DF 54 35 71 99 0A 09 D8 3D E4 61 BF A0 BE
    :   0A BE 11 A4 3C B5 38 41 41 48 04 E1 5B B1 17 1C
    :   53 B5 F4 C5 15 D3 FE 0C FB 0C AC EA 80 18 36 03
    :   7E 41 93 53 D7 40 74 49 DB D9 C6 AF FE D6 CA 0D
OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)

SEQUENCE {
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'CarlDSS'
  }
}

SEQUENCE {
  UTCTime '990816225050Z'
  UTCTime '991231235959Z'
}

SEQUENCE {
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'CarlDSS'
  }
}

SEQUENCE {
  INTEGER 00 B6 49 18 3E 8A 44 C1 29 71 94 4C 01 C4 12 C1
  7A 79 CB 54 4D AB 1E 81 FB C6 4C B3 0E 94 09 06
  EB 01 D4 B1 C8 71 4B C7 45 C0 50 25 5D 9C FC DA
  E4 6D D3 E2 86 48 84 82 7D BA 15 95 4A 16 F6 46
  ED DD F6 98 D2 BB 7E 8A 0A 8A BA 16 7B B9 50 01
  48 93 8B EB 25 15 51 97 55 DC 8F 53 0E 10 A9 50
  FC 70 B7 CD 30 54 FD DA DE A8 AA 22 B5 A1 AF 8B
  CC 02 88 E7 8B 70 5F B9 AD E1 08 D4 6D 29 2D D6
  E9
}

SEQUENCE {
  INTEGER 00 DD C1 2F DF 53 CE 0B 34 60 77 3E 02 A4 BF 8A
  5D 98 B9 10 D5
}

SEQUENCE {
  INTEGER 0C EE 57 9B 4B BD DA B6 07 6A 74 37 4F 55 7F 9D
  ED BC 61 0D EB 46 59 3C 56 0B 2B 5B 0C 91 CE A5
  62 52 69 CA E1 6D 3E BD BF FE E1 B7 B9 2B 61 3C
  AD CB AE 45 E3 06 AC 8C 22 9D 9C 44 87 0B C7 CD
  F0 1C D9 B5 4E 5D 73 DE AF OE C0 4C 1D 5A 51 F5 4F
  44 79 35 5A 73 AA 7F 46 51 1F A9 42 16 9C 48 EB
  8A 79 61 B4 D5 2F 53 22 44 63 1F 86 B8 A3 58 06
  25 F8 29 C0 EF BA E0 75 F0 42 C4 63 65 52 9B 0A
}

BIT STRING 0 unused bits
  02 81 81 00 99 87 74 27 03 66 A0 B1 C0 AD DC 2C
  75 BB E1 6C 44 9C DA 21 6D 4D 47 6D B1 62 09 E9
  D8 AE 1E F2 3A B4 94 B1 A3 8E 7A 9B 71 4E 00 94
  C9 B4 25 4E B9 60 96 19 24 01 F3 62 0C FE 75 C0
  FB CE D8 6B 00 E3 FD D5 70 4F DF 23 96 19 06 94
  F4 B1 61 8F 3A 57 B1 08 11 A4 0B 26 25 F0 52 76
  81 EA 0B 62 0D 95 2A E6 86 BA 72 B2 A7 50 83 0B

Schaad
CertDist
October 1999
[3] {
    SEQUENCE {
        SEQUENCE {
            OBJECT IDENTIFIER basicConstraints (2 5 29 19)
            BOOLEAN TRUE
            OCTET STRING
                30 03 01 01 FF
        }
        SEQUENCE {
            OBJECT IDENTIFIER keyUsage (2 5 29 15)
            BOOLEAN TRUE
            OCTET STRING
                03 02 01 86
        }
        SEQUENCE {
            OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
            OCTET STRING
                04 14 70 44 3E 82 2E 6F 87 DE 4A D3 75 E3 3D 20
                BC 43 2B 93 F1 1F
        }
        SEQUENCE {
            OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
        }
        SEQUENCE {
            INTEGER 2
            INTEGER 200
            SEQUENCE {
                OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
            }
        }
        SEQUENCE {
            BIT STRING 0 unused bits
                30 2D 02 14 6B A9 F0 4E 7A 5A 79 E3 F9 BE 3D 2B
                C9 06 37 E9 11 17 A1 13 02 15 00 8F 34 69 2A 8B
                B1 3C 03 79 94 32 4D 12 1F CE 89 FB 46 B2 3B
        }
        SEQUENCE {
            SEQUENCE {
                INTEGER 0
                INTEGER 2
            }
            SEQUENCE {
                INTEGER 200
            }
            SEQUENCE {
                SEQUENCE {
                    OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
                }
            }
            SEQUENCE {
                SEQUENCE {
                    SEQUENCE {
                        OCTET STRING 'CarlDSS'
                    }
                }
            }
            SEQUENCE {
                SEQUENCE {
                    SEQUENCE {
                        UTCTime '990817011049Z'
                    }
                }
            }
            SEQUENCE {
                SEQUENCE {
                    SEQUENCE {
                        UTCTime '391231235959Z'
                    }
                }
            }
        }
    }
}
SEQUENCE {
  OBJECT IDENTIFIER commonName (2 5 4 3)
  PrintableString 'AliceDSS'
}

SEQUENCE {
  OBJECT IDENTIFIER dsa (1 2 840 10040 4 1)
  INTEGER
    00 81 8D CD ED 83 EA 0A 9E 39 3E C2 48 28 A3 E4
    47 93 DD 0E D7 A8 0E EC 53 C5 AB 84 08 4F FF 94
    E1 73 48 7E 0C D6 F3 44 48 D1 FE 9F AF A4 A1 89
    2F E1 D9 30 C8 36 DE 3F 9B BF B7 4C DC 5F 69 8A
    E4 75 D0 37 0C 91 08 95 9B DE A7 5E F9 FC F4 9F
    2F DD 43 A8 8B 54 F1 3F B0 07 08 47 4D 5D 88 C3
    C3 B5 B3 E3 55 08 75 D5 39 76 10 C4 78 BD FF 9D
    B0 84 97 37 F2 E4 51 1B B5 E4 09 96 5C F3 7E 5B
    DB

Schaad
CertDist October 1999

SEQUENCE {
  INTEGER
    00 E2 47 A6 1A 45 66 B8 13 C6 DA 8F B8 37 21 2B
    62 8B F7 93 CD
}

SEQUENCE {
  INTEGER
    26 38 D0 14 89 32 AA 39 FB 3E 6D D9 4B 59 6A 4C
    76 23 39 04 02 35 5C F2 CB 1A 30 C3 1E 50 5D DD
    9B 59 E2 CD AA 05 3D 58 C0 7B A2 36 B8 6E 07 AF
    7D 8A 42 25 A7 F4 75 CF 4A 08 5E 4B 3E 90 F8 6D
    EA 9C C9 21 8A 3B 76 14 96 0B A9 7A DD E3 3F
    4F 79 6C 87 B7 13 11 34 A6 16 89 28 11 23
    D9 34 86 67 75 75 13 12 3D 43 5B 6F E5 51 BF FA
    89 F2 A2 1B 3E 24 7D 3D 07 8D 5B 63 C8 BB 45 A5
    A0 4A E3 85 D6 CE 06 80 3F E8 23 7E 1A F2 24 AB
    53 1A B8 27 0D 1E EF 08 BF 66 14 80 5C 62 AC 65
    FA 15 8B F1 BB 34 D4 D2 96 37 F6 61 47 B2 C4 32
    84 F0 7E 41 40 FD 46 A7 63 4E 33 F2 A5 E2 F4 F2
    83 E5 B8
}

SEQUENCE {
  BIT STRING 0 unused bits
    02 81 80 5C E3 B9 5A 75 14 96 0B A9 7A DD E3 3F
    A9 EC AC 5E DC BD B7 13 11 34 A6 16 89 28 11 23
    D9 34 86 67 75 75 13 12 3D 43 5B 6F E5 51 BF FA
    89 F2 A2 1B 3E 24 7D 3D 07 8D 5B 63 C8 BB 45 A5
    A0 4A E3 85 D6 CE 06 80 3F E8 23 7E 1A F2 24 AB
    53 1A B8 27 0D 1E EF 08 BF 66 14 80 5C 62 AC 65
    FA 15 8B F1 BB 34 D4 D2 96 37 F6 61 47 B2 C4 32
    84 F0 7E 41 40 FD 46 A7 63 4E 33 F2 A5 E2 F4 F2
    83 E5 B8
}

SEQUENCE {
  OCTET STRING
    01 1: BOOLEAN TRUE
    02: OCTET STRING
      30 00
2345 30 14:  SEQUENCE {
2347 06 3:   OBJECT IDENTIFIER keyUsage (2 5 29 15)
2352 01 1:   BOOLEAN TRUE
2355 04 4:   OCTET STRING
2359 02 6:   03 02 06 C0
2361 30 31:  SEQUENCE {
2363 06 3:   OBJECT IDENTIFIER
2366 04 14:   authorityKeyIdentifier (2 5 29 35)
2368 04 24:   OCTET STRING
2372 16 31:   30 16 80 14 70 44 3E 82 2E 6F 87 DE 4A D3 75 E3
2376 20 BC 43 2B 93 F1 1F
2394 30 29:  SEQUENCE {
2396 06 3:   OBJECT IDENTIFIER
2400 04 14:   subjectKeyIdentifier (2 5 29 14)
2401 04 22:   OCTET STRING
2405 14 BE 6C A1 B3 E3 C1 F7 ED 43 70 A4 CE 13 01
2409 26 FD E3 97 FE CD
2425 30 9:   SEQUENCE {
2427 06 7:   OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
2431 02 1:   INTEGER 2
2434 02 2:   INTEGER 201
2486 30 866: SEQUENCE {
2490 30 801: SEQUENCE {
2494 A0 3:   [0] {
2496 02 1:     INTEGER 2
2498 02 2:     INTEGER 201
2503 30 9:   SEQUENCE {
2505 06 7:   OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
2514 30 18: SEQUENCE {
2516 31 16:  SET {
2518 30 14:  SEQUENCE {
2520 06 3:   OBJECT IDENTIFIER commonName (2 5 4 3)
2525 13 7:   PrintableString ‘CarlDSS’
2534 30 30: SEQUENCE {
2536 17 13:  UTCTime ’990817011828Z’
2551 17 13:  UTCTime ’391231235959Z’
2566 30 16: SEQUENCE {
2568 31 14:  SET {
2570 30 12:  SEQUENCE {
2572 06 3:   OBJECT IDENTIFIER commonName (2 5 4 3)
2577 13 5:   PrintableString ‘bobDH’

SEQUENCE {
  dhPublicNumber (1 2 840 10046 2 1)
}
SEQUENCE {
  INTEGER
    00 EC 2C CD A4 EF 9A 26 2F 62 A7 BB 23 4D DF 2B
    25 C1 68 D2 9E A9 45 5B 36 F1 94 89 1A AF 7D 11
    24 9D 3D B9 3C 29 E8 D7 23 80 33 A6 9E 45 02 BB
    AA CC 9E 28 05 95 A0 B3 17 76 C1 F7 25 35 61 02
    41 92 27 0C 5E AE 48 E5 F3 6E 38 EF 91 D1 CF 37
    
  INTEGER
    FE 9A 40 97 C8 2D 35 9E 9D 93 C6 F8 15 AF 3F DA
    74 3A B7 C4 93 B5 B9 BB 76 6C 1F A8 7E BC 3A AA
    43 0A 81 64 FC 63 F0 7B 71 98 FA C0 38 79 10 1A
    33
}
INTEGER
    00 BA 0B D7 74 3D E7 34 E5 4C 13 A7 95 9B BB F1
    E4 61 37 08 FB 12 C7 FB 9C 91 77 06 99 35 F0 48
    24 96 33 12 01 7E 8D EC 0B F6 B2 C0 63 A7 15 C5
    5E 95 86 A2 73 C5 49 46 37 79 60 FD 77 05 09 48
    9B 70 8D 3C 05 F6 CE 44 2C 7F 7D 1B 2B 15 DD F3
    05 2F BE B5 20 8F 8D F9 B4 A0 45 74 2B F4 3B 9D
    42 62 34 27 27 81 8E 6F 0F 5E 62 85 89 CC ED 21
    C3 91 70 06 54 EE 70 A8 92 55 5B 6E 19 22 4D 62
    A7
}
INTEGER
    00 C3 AB 4A 30 79 B3 D3 97 4E CA F5 A2 7D C7 70
    A3 45 F3 B3 A2 86 05 D2 3E 49 F9 9F D9 0A B3 BE
    BD
}
INTEGER
    01 34 FE C2 33 48 EB F6 3B 97 D9 E4 97 A7 60 A5
    25 69 34 FB FD 46 2A D6 C9 C4 C5 F7 D6 F4 04 19
    8D 9D 39 BA 37 68 69 67 55 FB F2 6B OE 47 C5 5B
    0B 4B OE 1C 1A BB 7B 75 B7 AA C3 AA DE 2B 3A
    2A 8D 02 87 37 47 83 D7 31 B4 25 A8 AC BB 11 8A
    42 53 11 92 B6 69 E7 2E 90 C1 7A FC 87 F4 F6 D7
    1A
}
SEQUENCE {
  BIT STRING 0 unused bits
    B9 FF 1C 93 44 67 37 D1 B2 F8 57 9A 32 4A C9 4A
    FF 3B EC 1E
}
INTEGER 29

BIT STRING 0 unused bits
    02 81 80 6F D4 F6 CD 94 9A 6E AF 5B 57 17 96 75
    BB 0F B9 48 E9 90 37 0D 15 20 C2 55 1E 13 E2 AE
    71 17 84 C3 0E 74 AE 8A 55 7F 28 7D 8B D7 28 22
    9C 76 46 D7 3B 4F 9D D1 4D 1B B2 DB 51 94 C5 6D
    54 96 40 3B 8A 38 81 63 4A 8C C3 1E 09 89 74 A6
    58 D5 C8 5A 3D CF BB B8 23 7F 9C 1F 7D 78 FA 9E
    F9 90 9E 91 E7 4B C2 A4 BE 45 06 78 42 58 3D 9F
    63 2C EF 84 D4 67 E5 FB C6 6D A2 36 29 67 90 46
    DB 4E 48

Schaad

CertDist

October 1999
3166 A3 127:  [3] {
3168 30 125:    SEQUENCE {
3170 30 29:      SEQUENCE {
3172 06 3:        OBJECT IDENTIFIER subjectAltName (2 5 29 17)
3177 04 22:          OCTET STRING
:          30 14 81 12 62 6F 62 44 68 40 65 78 61 6D 70 65 73 2E 63 6F 6D
:          }
3201 30 12:      SEQUENCE {
3203 06 3:        OBJECT IDENTIFIER basicConstraints (2 5 29 19)
Schaad 17
CertDist  
October 1999
3208 01 1:        BOOLEAN TRUE
3211 04 2:          OCTET STRING
:          30 00
:          }
3215 30 14:      SEQUENCE {
3217 06 3:        OBJECT IDENTIFIER keyUsage (2 5 29 15)
3222 01 1:        BOOLEAN TRUE
3225 04 4:          OCTET STRING
:          03 02 03 08
:          }
3231 30 31:      SEQUENCE {
3233 06 3:        OBJECT IDENTIFIER
:          authorityKeyIdentifier (2 5 29 35)
3238 04 24:          OCTET STRING
:          30 16 80 14 70 44 3E 82 2E 6F 87 DE 4A D3 75 E3
:          3D 20 BC 43 2B 93 F1 1F
:          }
3264 30 29:      SEQUENCE {
3266 06 3:        OBJECT IDENTIFIER
:          subjectKeyIdentifier (2 5 29 14)
3271 04 22:          OCTET STRING
:          04 14 26 FF 19 48 C3 59 33 68 56 8D 7E C8 80 68
:          5C CF 3C 72 DD 26
:          }
:          }
:          }
3295 30 9:      SEQUENCE {
3297 06 7:        OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
:          }
3306 03 48:      BIT STRING 0 unused bits
:          30 2D 02 14 15 EA 15 43 E3 49 22 86 C1 BB E5 DA
:          E4 0E B8 09 E0 D5 72 35 02 15 00 AE 4F 51 29 73
:          71 75 A9 81 EB ED 9D 5E 00 19 7E F0 DE 5A D6
:          }
:          }
3356 31 283:    SET {
3360 30 279:      SEQUENCE {
3364 02 1:        INTEGER 1
3367 30 24:      SEQUENCE {
3369 30 18:      SEQUENCE {
3371 31 16:    SET {
3373 30 14:      SEQUENCE {
3375 06 3:        OBJECT IDENTIFIER commonName (2 5 4 3)
3380 13 7:      PrintableString 'CarlDSS'
:          }
:          }
:          }
:          }
:          }
:          }
:  
:  
3389 02 2: INTEGER 200
:  
3393 30 9: SEQUENCE {
3395 06 5: OBJECT IDENTIFIER sha1 {1 3 14 3 2 26}
3402 05 0: NULL
:  
3404 A0 176: [0] {
Schaad 18
CertDist October 1999

3407 30 26: SEQUENCE {
3409 06 9: OBJECT IDENTIFIER
: contentType {1 2 840 113549 1 9 3}
3420 31 13: SET {
3422 06 11: OBJECT IDENTIFIER
: id-ct-publishCert {1 2 840 113549 1 9 16 1 3}
:  
:  
3435 30 35: SEQUENCE {
3437 06 9: OBJECT IDENTIFIER
: messageDigest {1 2 840 113549 1 9 4}
3448 31 22: SET {
3450 04 20: OCTET STRING
: DA 39 A3 EE 5E 6B 4B 0D 32 55 BF EF 95 60 18 90
: AF D8 07 09
:  
:  
3472 30 109: SEQUENCE {
3474 06 11: OBJECT IDENTIFIER
: id-aa-smimeEncryptCerts {1 2 840 113549 1 9 16 2 13}
3487 31 94: SET {
3491 30 92: SEQUENCE {
3493 04 20: OCTET STRING
: 3B F6 B5 69 50 7E 3E AD 03 97 F8 F8 29 DD A0 B9
: 8A CF DA 9B
3515 30 12: SEQUENCE {
3517 30 10: SEQUENCE {
3519 06 8: OBJECT IDENTIFIER
: des-EDE3-CBC {1 2 840 113549 3 7}
:  
:  
3529 30 52: SEQUENCE {
3531 04 20: OCTET STRING
: E4 B8 2D 17 E4 23 D5 22 F0 58 BD 73 BD 3D 59 76
: AF C6 18 C8
3553 30 28: SEQUENCE {
3555 30 10: SEQUENCE {
3557 06 8: OBJECT IDENTIFIER
: des-EDE3-CBC {1 2 840 113549 3 7}
:  
:  
3567 30 14: SEQUENCE {
3569 06 8: OBJECT IDENTIFIER
: rc2CBC {1 2 840 113549 3 2}
3579 02 2: INTEGER 160
:  
:  

References


MUSTSHOULD Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997.


Security Considerations

This entire document discusses security. Some items of special note are:

Implementations must protect the signer’s private key. Compromise of the signer’s private key permits masquerading and therefore substitution of encryption certificates.

Implementations must do appropriate checking that the entity named in a certificate is the same entity that the encrypted message is destined for to protect contents of encrypted messages.

Author Address

Jim Schaad
Microsoft
One Microsoft Way
Redmond, WA 98052-6399
Jimsch@Microsoft.com