Certificate Distribution Specification
draft-ietf-smime-certdist-05.txt

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.

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.

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.

Public Directory

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. However, the use of directories has two drawbacks: First, the set of bulk algorithms supported by the recipient is unknown. Second, the chain of certificates needed to validate the user's certificate needs to be found in another manner.

Although there exists a property for listing bulk algorithms in the X.509 directory, it has no way of binding a list of algorithms to a single certificate. It is possible that a certificate bound to a key located on a hardware device is limited to a small set of algorithms, while a certificate bound to a software implementation can have a greater set of algorithms associated with it. The problem of determining what to publish is made all the harder because it is possible the intersection of the algorithms is empty. We therefore need to have a method that binds a specific list of algorithms to a specific certificate.

Building the necessary chain of certificates is the other problem. While it is possible to do direct lookup using an X.509 directory, the same is not true of an LDAP directory especially if one is using cross-certificates to a different root. While the problem is made somewhat easier by the Authority Information Access extension (it is possible to know where to look for the issuer certificate), it still requires multiple network accesses to build the certificate chain for what is relatively static information. If we can include at least one common chain with the user's certificate this problem is simplified.

Certificate Files

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).

Signed Objects

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 SignedData objects, this is the certificate bag in
the object. 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 be one instance 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-Schaad 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 3 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. The following auxiliary object class MAY be used to represent certificate subjects:

\[
\text{pkiUser OBJECT-CLASS ::= }
\begin{cases}
\text{SUBCLASS OF } \{ \text{top} \\
\text{KIND auxiliary} \\
\text{MAY CONTAIN } \{ \text{userSMimeCertificate} \\
\text{ID } \text{joint-iso-ccitt(2) ds(5) objectClass(6) pkiUser(21)}
\end{cases}
\]

\[
\text{userSMimeCertificate ATTRIBUTE ::= }
\begin{cases}
\text{WITH SYNTAX } \text{ContentInfo} \\
\text{EQUALITY MATCHING RULE contentInfoExactMatch} \\
\text{ID } 1 2 840 113549 1 9 16 4 1
\end{cases}
\]

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

\[
\text{SMimeCertDistributionSyntax}
\begin{cases}
\text{ISO(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) <TBD> }
\end{cases}
\]

\[
\text{DEFINITIONS IMPLICIT TAGS ::= }
\begin{cases}
\text{BEGIN}
\end{cases}
\]

-- 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.

Schaad 6
CertDist May 2000

IMPORTS
-- SMime Cryptographic Message Format
\[
\text{ContentInfo}
\begin{cases}
\text{FROM CryptographicMessageSyntax } \text{ISO(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms(1) }
\end{cases}
\]

-- SecureMimeMessageV3
\[
\text{SMIMECapabilities}
\begin{cases}
\text{FROM SecureMimeMessageV3 } \text{ISO(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) smime(4) }
\end{cases}
\]
-- S/MIME Object Identifier Registry
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.

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-SMimeEncryptionCert ::= IssuerAndSerialNumber

A description of IssuerAndSerialNumber can be found 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@exmsft.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)
Schaad                                   8
CertDist                                                   May 2000

  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
        :      }
      :    }
    :  }
  :}[0] {
    18 30 11:   OBJECT IDENTIFIER
          : id-ct-publishCert (1 2 840 113549 1 7 2 1)
48 A0 NDEF: [0] {
      :    }
    :  }
  :}[0] {
    18 30 11:   OBJECT IDENTIFIER
          : id-ct-publishCert (1 2 840 113549 1 9 16 1 3)
48 A0 NDEF: [0] {
      :    }
    :  }
  :}[0] {
    18 30 11:   OBJECT IDENTIFIER
          : id-ct-publishCert (1 2 840 113549 1 9 16 1 3)
48 A0 NDEF: [0] {
      :    }
    :  }
  :}[0] {
    18 30 11:   OBJECT IDENTIFIER
          : id-ct-publishCert (1 2 840 113549 1 9 16 1 3)
48 A0 NDEF: [0] {
      :    }
    :  }
  :}[0] {
68 02  1: INTEGER 2
    :   }
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)
115 13  7: PrintableString 'CarlRSA'
    :   }
    :   }
124 30 30: SEQUENCE {
126 17 13: UTCTime '990818070000Z'
141 17 13: UTCTime '391231235959Z'
    :   }
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
Schaad May 2000
CertDist

: 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 86 39 A7 BD D7 8D 1A 53 EC E4 00 D5
: E8 EC A2 36 B1 ED E2 50 E2 32 09 8A 3F 9F 99 25
: 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 {
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

SEQUENCE {
  OBJECT IDENTIFIER
  subjectKeyIdentifier (2 5 29 14)

OCTET STRING
  04 14 E9 E0 90 27 AC 78 20 7A 9A D3 4C F2 42 37
  4E 22 AE 9E 38 BB
}

SEQUENCE {
  OBJECT IDENTIFIER
  sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
  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
}

SEQUENCE {
  INTEGER 2
}

SEQUENCE {
  INTEGER
  0 unused bits
    46 34 6B C7 80 00 56 BC 11 D3 6E 2E CD 5D 71 D0
}

SEQUENCE {
  OBJECT IDENTIFIER
  sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
}

SEQUENCE {
  SET {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString ‘CarlRSA’
  }
}

SEQUENCE {
  UTCTime ‘990819070000Z’
  UTCTime ‘391231235959Z’
}

SEQUENCE {
  OBJECT IDENTIFIER commonName (2 5 4 3)
  PrintableString ‘BobRSA’
}

Schaad CertDist

Schaad May 2000
SEQUENCE {
  rsaEncryption (1 2 840 113549 1 1 1)
  NULL
}

BIT STRING 0 unused bits

SEQUENCE {
  basicConstraints (2 5 29 19)
  TRUE
  03 00
}

SEQUENCE {
  keyUsage (2 5 29 15)
  TRUE
  03 02 05 20
}

SEQUENCE {
  authorityKeyIdentifier (2 5 29 35)
  30 16 80 14 E9 E0 90 27 AC 78 20 7A 9A D3 4C F2
  42 37 4E 22 AE 9E 38 BB
}

SEQUENCE {
  subjectKeyIdentifier (2 5 29 14)
  30 16 80 14 E9 E0 90 27 AC 78 20 7A 9A D3 4C F2
  42 37 4E 22 AE 9E 38 BB
}

SEQUENCE {
  shalwithRSAEncryption (1 2 840 113549 1 1 5)
  NULL
}

BIT STRING 0 unused bits

CertDist

May 2000
SEQUENCE {
  [0] {
    INTEGER 2
  }
  INTEGER 1
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
    }
  }
  SEQUENCE {
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER commonName (2 5 4 3)
        PrintableString 'CarlDSS'
      }
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'CarlDSS'
    }
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'CarlDSS'
  }
  SEQUENCE {
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER dsa (1 2 840 10040 4 1)
        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 OE 10 A9 50
          FC 70 B7 CD 30 54 FD DA DE A8 A8 22 B5 A1 AF 8B
          CC 02 88 E7 8B 70 5F B9 AD E1 08 D4 6D 29 2D D6
          E9
        }
        INTEGER
        00 DD C1 2F DF 53 CE 0B 34 60 77 3E 02 A4 BF 8A
        5D 98 B9 10 D5
      }
      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 0E C9 1D 5A 51 F5 4F
    }
  }
}

Schaad
CertDist
May 2000
BIT STRING 0 unused bits

SEQUENCE {
  OBJECT IDENTIFIER basicConstraints (2 5 29 19)
  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 {
  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 {
  OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)

SEQUENCE {
  [0] {
    INTEGER 2
  }
  INTEGER 200
  SEQUENCE {
    OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
  }
  SEQUENCE {

Schaad
CertDist
May 2000
1778 31 16: SET {
1780 30 14: SEQUENCE {
1782 06 3: OBJECT IDENTIFIER commonName (2 5 4 3)
1787 13 7: PrintableString 'CarlDSS'
}
}

1796 30 30: SEQUENCE {
1798 17 13: UTCTime '990817011049Z'
1813 17 13: UTCTime '391231235959Z'
}

1828 30 19: SEQUENCE {
1830 31 17: SET {
1832 30 15: SEQUENCE {
1834 06 3: OBJECT IDENTIFIER commonName (2 5 4 3)
1839 13 8: PrintableString 'AliceDSS'
}
}

1849 30 438: SEQUENCE {
1853 30 299: SEQUENCE {
Schaad
CertDist
May 2000

1857 06 7: OBJECT IDENTIFIER dsa (1 2 840 10040 4 1)
1866 30 286: SEQUENCE {
1870 02 129: 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 A8
: 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 7C 76 7B 84 08 96 5C F3 7E 5B
: DB

2002 02 21: INTEGER
: 00 E2 47 A6 1A 45 66 B8 13 C6 DA 8F B8 37 21 2B
: 62 8B F7 93 CD

2025 02 128: 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 E9 C6 2E 5D A3 07 CD 23
: 85 B8 2F 30 01 7C 6D 49 89 11 89 36 44 BD F8 C8
: 95 4A 53 56 B5 E2 F9 73 EC 1A 61 36 1F 11 7F C2
: BD ED D1 50 FF 98 74 C2 D1 B1 4A 60 39 BA 36 39
}
}

2156 03 132: 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 OD 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
}
A3 131:       [3] {
2294 30 128:        SEQUENCE {
2297 30 32:           SEQUENCE {
2299 06 3:              OBJECT IDENTIFIER subjectAltName (2 5 29 17)
2304 04 25:              OCTET STRING
2307 01 30 17:                81 15 61 6C 69 63 65 44 73 73 40 65 78 61
2312 01 6D 70 6C 65 73 2E 63 6F 6D
2317 01 12:              SEQUENCE {
2320 06 3:              OBJECT IDENTIFIER basicConstraints (2 5 29 19)
2325 01 1:              BOOLEAN TRUE
2328 04 2:              OCTET STRING
2331 01 30 00
2334 01 14:              SEQUENCE {
2337 06 3:              OBJECT IDENTIFIER keyUsage (2 5 29 15)
2342 01 1:              BOOLEAN TRUE
2345 04 4:              OCTET STRING
Schaad
2348 04 CertDist
2351 01 15
May 2000
2354 01 03 02 06 C0
2357 01 03
2360 01 30 31:        SEQUENCE {
2363 06 3:              OBJECT IDENTIFIER
2366 04 24:              OCTET STRING
2369 01 30 16:                80 14 70 44 3E 82 2E 6F 87 DE 4A D3 75 E3
2374 01 3D 20 BC 43 2B 93 F1 1F
2377 01 29:              SEQUENCE {
2380 06 3:              OBJECT IDENTIFIER
2383 04 22:              OCTET STRING
2386 01 04 14 BE 6C A1 B3 E3 C1 F7 ED 43 70 A4 CE 13 01
2391 01 E2 FD E3 97 FE CD
2394 01 :              }
2397 01 :              }
2400 01 :              }
2403 01 :              }
2406 01 :              }
2409 01 :              }
2412 01 :              }
2415 01 :              }
2418 01 :              }
2421 01 :              }
2424 01 :              }
2427 01 :              }
2430 01 :              }
2433 01 03 02 15 00 98 B0 C6 3F CF 71 47 5A 35 A9 4A
2436 01 8F C0 F8 24 05 E8 46 94 8E 02 14 5B 9F 48 C0 8C
2439 01 A1 C1 02 9C 44 EA E9 A1 87 C1 A5 7F 28 2D BB
2442 01 :
2445 01 :
2448 01 :
2451 01 :
2454 01 :
2457 01 :
2460 01 :
2463 01 :
2466 01 :
2469 01 :
2472 01 :
2475 01 :
2478 01 :
2481 01 :
2484 01 :
2487 01 :
2490 01 :
2493 01 :
2496 01 02 1:              INTEGER 2
2499 01 02 2:              INTEGER 201
2502 01 30 9:              SEQUENCE {
2505 06 7:              OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
2508 01 18:              SEQUENCE {
2511 31 16:              SET {
2514 30 14:              SEQUENCE {
2517 46 3:              OBJECT IDENTIFIER commonName (2 5 4 3)
PrintableString "CarlDSS"

UTCTime '990817011828Z'

UTCTime '391231235959Z'

PrintableString 'bobDH'

dhPublicNumber (1 2 840 10046 2 1)

INTEGER

BIT STRING 0 unused bits
3031 03 132:  BIT STRING 0 unused bits
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
3179 30 14 81 12 62 6F 62 44 68 40 65 78 61 6D 70 6C
3185 30 84 6F 62 44 68 40 65 78 61 6D 73 2E 63 6F 6D
3201 30 12:    SEQUENCE {
3203 06 3:      OBJECT IDENTIFIER basicConstraints (2 5 29 19)
3208 01 1:      BOOLEAN TRUE
3211 04 2:      OCTET STRING
3213 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
3227 30 03 02 03 08
3231 30 31:    SEQUENCE {
3233 06 3:      OBJECT IDENTIFIER
3235 04 24:      OCTET STRING
3237 30 16 80 14 70 44 3E 82 2E 6F 87 DE 4A D3 75 E3
323D 3D 20 BC 43 2B 93 F1 1F
3264 30 29:    SEQUENCE {
3266 06 3:      OBJECT IDENTIFIER
3268 04 22:      OCTET STRING
3270 04 14 26 FF 19 48 C3 59 33 68 56 8D 7E C8 80 68
3272 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

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