Example call flows using Session Initiation Protocol (SIP) security mechanisms
draft-ietf-sip-sec-flows-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 17, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document shows example call flows demonstrating the use of Transport Layer Security (TLS), and Secure/Multipurpose Internet Mail Extensions (S/MIME) in Session Initiation Protocol (SIP). It also provides information that helps implementers build interoperable SIP
software. To help facilitate interoperability testing, it includes certificates used in the example call flows and processes to create certificates for testing.

Table of Contents

1. Administrative Information ........................................... 3
2. Introduction .......................................................... 3
3. Conventions ............................................................ 4
4. Security Considerations ................................................ 4
5. Known Problems .......................................................... 4
6. Certificates ............................................................. 4
   6.1. CA Certificates ..................................................... 4
   6.2. Host Certificates ................................................... 8
   6.3. User Certificates .................................................... 9
7. Callflow with Message Over TLS ........................................ 12
   7.1. TLS with Server Authentication .................................... 12
   7.2. MESSAGE Message Over TLS ......................................... 13
8. Callflow with S/MIME-secured Message ................................. 14
   8.1. MESSAGE Message with Signed Body ................................. 14
   8.2. MESSAGE Message with Encrypted Body .............................. 17
   8.3. MESSAGE Message with Encrypted and Signed Body .................. 20
9. Test Consideration ..................................................... 25
10. IANA Considerations .................................................. 26
11. Acknowledgments ....................................................... 26
12. References ............................................................ 26
   12.1. Normative References ............................................. 26
   12.2. Informative References ............................................ 27
Appendix A. Making Test Certificates .................................... 27
   A.1. makeCA script ..................................................... 29
   A.2. makeCert script ................................................... 31
Appendix B. Certificates for Testing .................................... 33
Appendix C. Message Dumps ............................................... 36
Authors’ Addresses ....................................................... 44
Intellectual Property and Copyright Statements .......................... 45
1. Administrative Information

Please note that this version of the document (draft-ietf-sip-sec-flows-00) is ONLY an administrative renaming from draft-ietf-sipping-sec-flows. The outstanding additions to this document discussed in those working groups have not yet been captured here.

2. Introduction

Several different groups are starting to implement the S/MIME[7] portion of SIP[2], and SIP with TLS[4], implementations are becoming very common. At several interoperability events, it has become clear that it is difficult to write these systems without any test vectors or examples of "known good" messages to test against. Furthermore, testing at the events is often hampered by trying to get certificates signed by some common test root into the appropriate format for various clients. This document addresses both of these issues by providing messages that give detailed examples that implementers can use for comparison and that can also be used for testing. In addition, this document provides a common certificate that can be used for a Certificate Authority (CA) to reduce the time it takes to set up a test at an interoperability event. The document also provides some hints and clarifications for implementers.

A simple SIP call flow using SIPS URIs and TLS is shown in Section 7. The certificates for the hosts used are shown in Section 6.2, and the CA certificates used to sign these are shown in Section 6.1.

The text from Section 8.1 through Section 8.3 shows some simple SIP call flows using S/MIME to sign and encrypt the body of the message. The user certificates used in these examples are shown in Section 6.3. These host certificates are signed with the same CA certificate.

Section 9 presents a partial list of things implementers should consider in order to implement systems that will interoperate.

A way to make certificates that can be used for interoperability testing is presented in Appendix A, along with methods for converting these to various formats.

Binary copies of various messages in this draft that can be used for testing appear in Appendix C.
3. Conventions

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 [1].

4. Security Considerations

Implementers must never use any of the certificates provided in this document in anything but a test environment. Installing the CA root certificates used in this document as a trusted root in operational software would completely destroy the security of the system while giving the user the impression that the system was operating securely.

This document recommends some things that implementers might test or verify to improve the security of their implementations. It is impossible to make a comprehensive list of these, and this document only suggests some of the most common mistakes that have been seen at the SIPit interoperability events. Just because an implementation does everything this document recommends does not make it secure.

This document does not show the messages to check Certificate Revocation Lists (see [3]) as that is not part of the SIP call flow.

5. Known Problems

The binary dumps of the messages in Section 7.2 need to be updated to match the text of the draft.

The messages are missing the accept headers. They should have the following header:

Accept: multipart/signed
Accept: text/plain
Accept: application/pkcs7-mime
Accept: application/sdp
Accept: multipart/alternative

6. Certificates

6.1. CA Certificates

The certificate used by the CA to sign the other certificates is shown below. This is a X509v3 certificate. Note that the basic
Internet-Draft            SIP Secure Call Flows                 May 2006

constraints allow it to be used as a CA.

Version: 3 (0x2)
Serial Number: 0 (0x0)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
OU=Sipit Test Certificate Authority
Validity
Not Before: Jul 18 12:21:52 2003 GMT
Not After : Jul 15 12:21:52 2013 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
OU=Sipit Test Certificate Authority
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
  00:c3:22:1e:83:91:c5:03:2c:3c:8a:f4:11:14:c6:
  07:4d:52:b0:f8:37:7b:e2:0a:27:30:70:dd:ff:9e:2e:
  a3:ff:1b:d0:68:28:e1:9d:e5
  Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:
  DirName:/C=US/ST=California/L=San Jose/O=sipit/
  OU=Sipit Test Certificate Authority
  serial:00
X509v3 Basic Constraints:
  CA:TRUE
Signature Algorithm: sha1WithRSAEncryption
  6f:52
The ASN.1 parse of the CA certificate is shown below.

```
0:l= 804 cons: SEQUENCE
 4:l= 653 cons: SEQUENCE
 8:l=  3 cons:  cont [ 0 ]
10:l=  1 prim:  INTEGER       :02
13:l=  1 prim:  INTEGER       :00
16:l=  13 cons:  SEQUENCE
18:l=  9 prim:  OBJECT        :sha1WithRSAEncryption
29:l=  0 prim:  NULL
31:l= 112 cons:  SEQUENCE
33:l=  11 cons:  SET
35:l=  9 cons:  SEQUENCE
37:l=  3 prim:  OBJECT        :countryName
42:l=  2 prim:  PRINTABLESTRING :US
46:l=  19 cons:  SET
48:l= 17 cons:  SEQUENCE
50:l=  3 prim:  OBJECT        :stateOrProvinceName
55:l= 10 prim:  PRINTABLESTRING :California
67:l= 17 cons:  SET
69:l=  15 cons:  SEQUENCE
71:l=  3 prim:  OBJECT        :localityName
76:l=  8 prim:  PRINTABLESTRING :San Jose
86:l= 14 cons:  SET
88:l= 12 cons:  SEQUENCE
90:l=  3 prim:  OBJECT        :organizationName
95:l=  5 prim:  PRINTABLESTRING :sipit
102:l= 41 cons:  SET
104:l= 39 cons:  SEQUENCE
106:l=  3 prim:  OBJECT        :organizationalUnitName
111:l= 32 prim:  PRINTABLESTRING :
              Sipit Test Certificate Authority
145:l= 30 cons:  SEQUENCE
147:l=  13 prim:  UTCTIME      :030718122152Z
162:l=  13 prim:  UTCTIME      :130715122152Z
177:l= 112 cons:  SEQUENCE
179:l=  11 cons:  SET
181:l=  9 cons:  SEQUENCE
183:l=  3 prim:  OBJECT        :countryName
188:l=  2 prim:  PRINTABLESTRING :US
192:l=  19 cons:  SET
194:l=  17 cons:  SEQUENCE
196:l=  3 prim:  OBJECT        :stateOrProvinceName
201:l=  10 prim:  PRINTABLESTRING :California
213:l=  17 cons:  SET
215:l=  15 cons:  SEQUENCE
```
217:l=3 prim: OBJECT :localityName
222:l=8 prim: PRINTABLESTRING :San Jose
232:l=14 cons: SET
234:l=12 cons: SEQUENCE
236:l=3 prim: OBJECT :organizationName
241:l=5 prim: PRINTABLESTRING :sipit
248:l=14 cons: SET
250:l=39 cons: SEQUENCE
252:l=3 prim: OBJECT :organizationalUnitName
257:l=32 prim: PRINTABLESTRING :Sipit Test Certificate Authority
291:l=159 cons: SEQUENCE
294:l=13 cons: SEQUENCE
296:l=9 prim: OBJECT :rsaEncryption
307:l=0 prim: NULL
309:l=141 prim: BIT STRING
00 30 81 89 02 81 81 00-c3 22 1e 83 91 c5 03 2c .0.......".....,
3c 8a f4 11 14 c6 4b 9d-fa 72 78 c6 b0 95 18 a7 <.......K..rx.....
e0 8c 79 ba 5d a4 ae 1e-21 2d 9d f1 0b 1c cf bd y].]...1--------
5b 29 b3 90 13 73 66 92-6e df 4c b3 b3 1c 1f 2a []...sf.n.L....*
82 0a ba 07 4d 52 b0 f8-37 7b e2 0a 27 30 70 dd ....MR..7('p.
f9 2e 03 ff 2a 76 cd df-87 1a bd 71 eb e1 99 6a ...."v......q...j
c4 7f 8e 74 a0 77 85 04-e9 41 ad fc 03 b6 17 75 ...t.w.A......u
aa 33 ea 0a 16 d9 fb 79-32 2e f8 cf 4d c6 34 a3 ...3...y2...M.4.
ff 1b d0 68 28 e1 9d e5-02 03 01 00 01 ...h(.......453:l=205 cons: cont [ 3 ]
456:l=202 cons: SEQUENCE
459:l=29 cons: SEQUENCE
461:l=3 prim: OBJECT :X509v3 Subject Key Identifier
466:l=22 prim: OCTET STRING
04 14 6b 46 17 14 ea 94-76 25 80 54 6e 13 54 da ..kf....v%.Tn.T.
a1 e3 54 14 a1 b6 ..T...
490:l=154 cons: SEQUENCE
493:l=3 prim: OBJECT :X509v3 Authority Key Identifier
498:l=146 prim: OCTET STRING
30 81 8f 80 14 6b 46 17-14 ea 94 76 25 80 54 6e 0....kf....v%.Tn
13 54 da a1 e3 54 14 a1-b6 a1 74 a4 47 32 70 30 31 .T...t...r0pl
0b 30 09 06 03 55 04 06-13 02 55 53 31 13 30 11 .0...U...US1.0.
06 03 55 04 08 13 0a 43-61 6c 69 66 6f 72 6e 69 ...U....California
61 31 11 30 0f 06 03 55-04 07 13 08 53 61 6e 20 a1.0...U....San
4a 6f 73 65 31 0e 30 0c-06 03 55 04 a1 30 05 73 Jose1.0...U....s
69 70 69 74 31 29 30 27-06 03 55 04 0b 13 20 53 ipit1)0'...U... S
69 70 69 74 20 54 65 73-74 72 20 43 65 65 72 69 66 ipit Test Certif
69 63 61 74 65 20 41 75-74 75 68 6f 72 69 74 79 82 icate Authority.
01 .

0092 - <SPACES/NULS>

647:l=12 cons: SEQUENCE
649:l=3 prim: OBJECT :X509v3 Basic Constraints
6.2. Host Certificates

The certificate for the host example.com is shown below. Note that the Subject Alternative Name is set to example.com and is a DNS type. The certificates for the other hosts are shown in Appendix B.
Data:

Version: 3 (0x2)
Serial Number:
01:95:00:71:02:33:00:55
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
OU=Sipit Test Certificate Authority
Validity
Not Before: Feb 3 18:49:08 2005 GMT
Not After : Feb 3 18:49:08 2008 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
CN=example.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
3b:4a:aaf:f2:ce:d1:0c:bc:c0:5f:31:6a:43:e7:7c:
5d:2d:64:85:b1:a4:ca:01:f1:8e:c5:7e:0f:ff:00:
91:a3:ea:cb:3e:12:02:75:a4:bb:08:c8:d0:2a:ef:
Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Alternative Name:
    DNS:example.com
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Subject Key Identifier:
    44:B0:77:65:8D
Signature Algorithm: sha1WithRSAEncryption

6.3. User Certificates

The user certificate for fluffy@example.com is shown below. Note
that the Subject Alternative Name has a list of names with different
URL types such as a sip, im, or pres URL. This is necessary for interoperating with CPIM gateway. In this example, example.com is the domain for fluffy. The message could be coming from a host called atlanta.example.com, and the AOR in the user certificate would still be the same. The others are shown in Appendix B.
Data:
Version: 3 (0x2)
Serial Number:
01:95:00:71:02:33:00:58
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
OU=Sipit Test Certificate Authority
Validity
Not Before: Feb 3 18:49:34 2005 GMT
Not After : Feb 3 18:49:34 2008 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
CN=fluffy@example.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:ca:ab:9b:9b:4e:3c:d5:45:3c:ce:00:a6:36:a8:
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Alternative Name:
URI:sip:fluffy@example.com, URI:im:fluffy@example.com,
URI:pres:fluffy@example.com
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Key Identifier:
7E:95:70
Signature Algorithm: sha1WithRSAEncryption
de:0a:e2:ee:ab9:f0:5c:9f:ad:0a:de:e5:4e:21:8f:11:d8:41:
3d:ff
7. Callflow with Message Over TLS

7.1. TLS with Server Authentication

The flow below shows the edited SSLDump output of the host example.com forming a TLS[4] connection to example.net. In this example mutual authentication is not used. Note that the client proposed three protocol suites including TLS_RSA_WITH_AES_128_CBC_SHA defined in [6]. The certificate returned by the server contains a Subject Alternative Name that is set to example.net. A detailed discussion of TLS can be found in [13].

This example does not use the Server Extended Hello[5].

New TCP connection #1: 127.0.0.1(55768) <-> 127.0.0.1(5061)
1 1 0.0060 (0.0060) C>SV3.1(49) Handshake
   ClientHello
      Version 3.1
      random[32]=
         42 16 8c c7 82 cd c5 87 42 ba f5 1c 91 04 fb 7d
         4d 6c 56 f1 db 1d ce 8a b1 25 71 5a 68 01 a2 14
      cipher suites
         TLS_RSA_WITH_AES_256_CBC_SHA
         TLS_RSA_WITH_AES_128_CBC_SHA
         TLS_RSA_WITH_3DES_EDE_CBC_SHA
      compression methods
         NULL
1 2 0.0138 (0.0077) S>CV3.1(74) Handshake
   ServerHello
      Version 3.1
      random[32]=
         42 16 8c c7 c9 2c 43 42 bb 69 a5 ba f1 2d 69 75
         c3 8d 3a 85 78 19 f2 e4 d9 2b 72 b4 cc dd e4 72
      session_id[32]=
         06 37 e9 22 56 29 e6 b4 3a 6e 53 fe 56 27 ed 1f
         2a 75 34 65 f0 91 fc 79 cf 90 da ac f4 6f 64 b5
      cipherSuite TLS_RSA_WITH_AES_256_CBC_SHA
      compressionMethod NULL
1 3 0.0138 (0.0000) S>CV3.1(1477) Handshake
   Certificate
1 4 0.0138 (0.0000) S>CV3.1(4) Handshake
   ServerHelloDone
1 5 0.0183 (0.0045) C>SV3.1(134) Handshake
   ClientKeyExchange
      EncryptedPreMasterSecret[128]=
         a6 bd d9 4b 76 4b 9d 6f 7b 12 8a e4 52 75 9d 74
         4f 06 e4 b0 bc 69 96 d7 42 ba 77 01 b6 9e 64 b0
Once the TLS session is set up, the following MESSAGE message (as defined in [12]) is sent from fluffy@example.com to kumiko@example.net. Note that the URI has a SIPS URL and that the VIA indicates that TLS was used. In order to format this document, it was necessary to break up some of the lines across continuation lines but the original messages have no continuation lines and no breaks in the Identity header field value.

```
MESSAGE sips:kumiko@example.net SIP/2.0
To:     <sips:kumiko@example.net>
From:   <sips:fluffy@example.com>;tag=03de46e1
Via:    SIP/2.0/TLS 127.0.0.1:5071;
          branch=z9hG4bK-d87543-58c826887160f95f-1--d87543--rport
Call-ID: 0dc68373623af98a@Y2ouY2lzY28uc2lwaXQubmV0
CSeq:   1 MESSAGE
Contact: <sips:fluffy@127.0.0.1:5071>
Max-Forwards: 70
Content-Transfer-Encoding: binary
Content-Type: text/plain
Date:     Sat, 19 Feb 2005 00:48:07 GMT
User-Agent: SIPimp.org/0.2.5 (curses)
Content-Length: 6

Hello!
```

The response is sent from example.net to example.com over the same TLS connection. It is shown below.
SIP/2.0 200 OK
To: <sips:kumiko@example.net>;tag=4c53f1b8
From: <sips:fluffy@example.com>;tag=03de46e1
Via: SIP/2.0/TLS 127.0.0.1:5071;
    branch=z9hG4bK-d87543-58c826887160f95f-1--d87543--
    rport=55768;received=127.0.0.1
Call-ID: 0dc68373623af98a@Y2ouY2lzY28uc2lwaXQubmV0
CSeq: 1 MESSAGE
Contact: <sips:kumiko@127.0.0.1:5061>
Content-Length: 0

8. Callflow with S/MIME-secured Message

8.1. MESSAGE Message with Signed Body

Example Signed Message. The value on the Content-Type line has been broken across lines to fit on the page but it should not be broken across lines in actual implementations.
It is important to note that the signature is computed across includes the header and excludes the boundary. The value on the Message-body line ends with CRLF. The CRLF is included in the boundary and should not be part of the signature computation. In the example below, the signature is computed over data starting with the C in the Content-Type and ending with the o in the hello.

ASN.1 parse of binary Blob 1. Note that at address 30, the hash for the signature is specified as SHA1. Also note that the sender’s certificate is not attached as it is optional in [8].
4:  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15:  [0] {
19:    SEQUENCE {
23:      INTEGER 1
26:      SET {
28:        SEQUENCE {
30:          OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
32:        }
37:      SEQUENCE {
39:        OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
41:      }
50:      SET {
54:        SEQUENCE {
58:          INTEGER 1
61:        SEQUENCE {
63:          SEQUENCE {
65:            SET {
67:              SEQUENCE {
69:                OBJECT IDENTIFIER countryName (2 5 4 6)
71:                  PrintableString 'US'
73:              }
78:            SET {
80:              SEQUENCE {
82:                OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
84:                  PrintableString 'California'
86:              }
91:            }
95:          SET {
97:            SEQUENCE {
99:              OBJECT IDENTIFIER localityName (2 5 4 7)
101:            PrintableString 'San Jose'
103:            }
106:          }
109:        }
111:      SET {
113:        SEQUENCE {
115:          OBJECT IDENTIFIER organizationName (2 5 4 10)
117:            PrintableString 'sipit'
119:          }
122:        }
124:      }
128:    }
132:  }
135:  SEQUENCE {
136:    OBJECT IDENTIFIER
138:      organizationalUnitName (2 5 4 11)
140:      PrintableString
142:        'Sipit Test Certificate Authority'
144:    }

8.2. MESSAGE Message with Encrypted Body

Example encrypted text/plain message that says "hello":

```
C4 0E 40 A5 7F 88 5B 06 90 E7 B2 40 39 DF 33 E3
C4 0E 40 A5 7F 88 5B 06 90 E7 B2 40 39 DF 33 E3
```

:     )
:     )
177:   INTEGER 01 95 00 71 02 33 00 58
:     )
187:   SEQUENCE {
189:     OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
:     )
196:   SEQUENCE {
198:     OBJECT IDENTIFIER rsaEncryption
:     (1 2 840 113549 1 1 1)
209:   NULL
:     )
211:   OCTET STRING
:     C4 0E 40 A5 7F 88 5B 06 90 E7 B2 40 39 DF 33 E3
:     18 39 C2 9E EC 51 5E 06 E2 D5 DA F0 F6 87 77 1E
:     F7 F9 C1 26 04 20 F8 30 B8 C0 37 92 F6 5C 64 DD
:     87 41 43 F8 2D E5 28 20 35 7D 84 72 2B 5E 5F CF
:     2E 73 93 03 4B DB 35 4C CA 44 CD F8 91 58 A2 4C
:     65 A1 A6 EA DC E6 1B 1E DD DA BD BE 1A EA 9F 62
:     12 7A D1 1A E7 27 B5 96 88 B9 E6 EF 79 C0 E5 40
:     A0 5F 9F 93 09 4C 65 55 DA A8 FE CD 02 10 A9 67
:     )
:     )
:     )
:     )
MESSAGE sip:kumiko@example.net SIP/2.0
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=6d2a39e4
Via: SIP/2.0/UDP 68.122.119.3:5060;
branch=z9hG4bK-d87543-44ddc0a217a51788-1--d87543--;rport
Call-ID: 031be67669ea97999@Y2ouY2lzY28uc21waXQubmV0
CSeq: 1 MESSAGE
Contact: <sip:fluffy@68.122.119.3:5060>
Max-Forwards: 70
Content-Disposition: attachment; handling=required; filename=smime.p7
Content-Transfer-Encoding: binary
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Date: Sat, 19 Nov 2005 23:33:18 GMT
User-Agent: SIPimp.org/0.2.5 (curses)
Content-Length: 435

*****************
* BINARY BLOB 2 *
*****************

ASN.1 parse of binary Blob 2. Note that at address 324, the encryption is set to aes128-CBC.

    0:SEQUENCE {
        4:  OBJECT IDENTIFIER envelopedData (1 2 840 113549 1 7 3)
        15:  [0] {
            19:  SEQUENCE {
                23:  INTEGER 0
                26:  SET {
                    30:  SEQUENCE {
                        34:  INTEGER 0
                        37:  SEQUENCE {
                            39:  SEQUENCE {
                                41:  SET {
                                    43:  SEQUENCE {
                                        45:  OBJECT IDENTIFIER countryName (2 5 4 6)
                                        50:  PrintableString 'US'
                                        :   }
                                        :   }
                                    54:  SET {
                                        56:  SEQUENCE {
                                            58:  OBJECT IDENTIFIER stateOrProvinceName(2 5 4 8)
                                            63:  PrintableString 'California'
                                            :   }
                                            :   }
                                        75:  SET {

SEQUENCE {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    PrintableString 'San Jose'
}

SET {
    SEQUENCE {
        OBJECT IDENTIFIER organizationName (2 5 4 10)
        PrintableString 'sipit'
    }
}

SET {
    SEQUENCE {
        OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
        PrintableString 'Sipit Test Certificate Authority'
    }
}

INTEGER 01 95 00 71 02 33 00 57

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

OCTET STRING
    7C F3 8A 02 E8 44 2C A6 9B 3E 64 46 06 D3 95 2D
    DF 19 8F 5D 0C 24 6B F7 93 03 E7 3C 98 F1 57 74
    67 70 0E 40 F8 05 96 34 06 36 97 61 5C 0B 2D 61
    AD CB F0 82 56 23 E5 09 C0 C7 BC A5 F4 A3 B7 59
    5D 8B 44 6E 3F 7C DE 50 54 2C 95 73 CC 9A 74 8B
    A9 26 68 FD F8 82 01 43 1D 30 3C 0C 40 B2 19 A2
    5A 90 06 0F AC 95 CB DF 21 13 F2 26 C8 10 45 A3
    F4 AB 54 74 72 FD 91 6C 73 27 BF 62 47 7B EC 58

SEQUENCE {
    OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
}

SEQUENCE {
    OBJECT IDENTIFIER aes128-CBC (2 16 840 1 101 3 4 1 2)
}

OCTET STRING
    50 9E 44 AA A5 54 C3 5C 0D 9A DF 65 F7 47 36 99

[0]
    55 C5 C7 EA 5D 5A 7C 06 95 3C 24 25 D5 53 08 BB
    04 19 B4 BF 84 15 F5 6C 4C 80 05 14 06 3E F3 D1
    B7 04 A1 46 4E E3 1E FF 16 35 79 2A 06 DD A8 83
8.3. MESSAGE Message with Encrypted and Signed Body

In the example below, one of the headers is contained in a box and is split across two lines. This was only done to make it fit in the RFC format. This header should not have the box around it and should be on one line with no whitespace between the "mime;" and the "smime-type". Note that Content-Type is split across lines for formatting but is not split in the real message.
MESSAGE sip:kumiko@example.net SIP/2.0
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=361300da
Via: SIP/2.0/UDP 68.122.119.3:5060;
   branch=z9hG4bK-d87543-0710dbfb18ebb8e6-1--d87543--;rport
Call-ID: 5eda27a67de6283d@Y2ouY2lzY28uc21waXQubmV0
CSeq: 1 MESSAGE
Contact: <sip:fluffy@68.122.119.3:5060>
Max-Forwards: 70
Content-Transfer-Encoding: binary
Content-Type: multipart/signed;boundary=1af019eb7754ddf7;
   micalg=sha1;protocol="application/pkcs7-signature"
Date: Sat, 19 Nov 2005 23:35:40 GMT
User-Agent: SIPimp.org/0.2.5 (curses)
Content-Length: 1191

--1af019eb7754ddf7
<p>|--See note about stuff in this box -------------------------|
|Content-Type: application/pkcs7-mime;|</p>
<table>
<thead>
<tr>
<th>smime-type=enveloped-data;name=smime.p7m</th>
</tr>
</thead>
</table>
Content-Disposition: attachment;handling=required;filename=smime.p7
Content-Transfer-Encoding: binary

***************
* BINARY BLOB 3 *
***************
--1af019eb7754ddf7
Content-Type: application/pkcs7-mime;name=smime.p7s
Content-Disposition: attachment;handling=required;filename=smime.p7s
Content-Transfer-Encoding: binary

***************
* BINARY BLOB 4 *
***************
--1af019eb7754ddf7--

Binary blob 3

0:SEQUENCE {
   4: OBJECT IDENTIFIER envelopedData (1 2 840 113549 1 7 3)
   15: [0] {
      19: SEQUENCE {
         23: INTEGER 0
         26: SET {
            30: SEQUENCE {
               34: INTEGER 0

37:    SEQUENCE {
39:      SEQUENCE {
41:        SET {
43:          SEQUENCE {
45:            OBJECT IDENTIFIER countryName (2 5 4 6)
50:              PrintableString 'US'
54:          SET {
56:            SEQUENCE {
58:              OBJECT IDENTIFIER stateOrProvinceName(2 5 4 8)
63:              PrintableString 'California'
68:          SET {
72:            SEQUENCE {
76:              OBJECT IDENTIFIER localityName (2 5 4 7)
81:              PrintableString 'San Jose'
86:          SET {
90:            SEQUENCE {
94:              OBJECT IDENTIFIER organizationName (2 5 4 10)
99:              PrintableString 'sipit'
104:          SET {
108:            SEQUENCE {
112:              OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
117:              PrintableString 'Sipit Test Certificate Authority'
122:          SET {
126:            SEQUENCE {
130:              OBJECT IDENTIFIER rsaEncryption(1 2 840 113549 1 1 1)
135:              NULL
140:            OCTET STRING
145:              69 B3 A3 61 F4 F8 63 4F 46 0A 1A AB 0F 1B 16 09
150:              DB 3A A9 12 3B 23 F0 C9 4E 68 04 15 AB 42 4F 66
155:              FA EF 8D C4 86 88 41 BA 53 A3 88 49 54 E3 0E EB
160:              E3 69 63 5A DF 77 2A 8A 1E 42 7E E4 A7 DB CF 90
170:              53 D8 FC D9 70 9F 02 0F F2 D2 CB F7 15 7F 6F 4F
AB 19 0F 55 51 A2 76 24 DA A3 78 F4 1E 31 AA 6A
: DF 7C E2 42 3B C5 33 11 E0 EE EE 2E 02 9D 8C 1A
:
:
:
:

309: SEQUENCE {
311:   OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
322:   SEQUENCE {
324:     OBJECT IDENTIFIER aes128-CBC (2 16 840 1 101 3 4 1 2)
335:     OCTET STRING
: 72 71 AE FE 55 12 BA 99 92 EA D3 C5 9C B6 60 69
: [0]
353:   [0]
: 9A 9F DD 9E 58 B6 BE 59 BC CA 6C 3E 3E F5 81 A3
: 30 A0 38 A3 1C 25 92 E3 AA 07 7A 85 7C 36 F0 12
: 9F 80 DF 9B BD 1E 22 EC BF 8B 03 EB 33 AE 81 75
: D3 91 0A 82 1E 13 8C 60 F0 2B 55 DD 03 52 84 52
: B1 51 5F E2 F0 CE 8A 9A 4B F5 46 CE BF 77 80 8F
: }
: }
: }

Binary Blob 4

0:SEQUENCE {
  4: OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15: [0] {
19: SEQUENCE {
23:   INTEGER 1
26:   SET {
28:     SEQUENCE {
30:       OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
: }
: }
37: SEQUENCE {
39:   OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
: }
50: SET {
54: SEQUENCE {
58:   INTEGER 1
61:   SEQUENCE {
63:     SEQUENCE {
65:       SET {
67:         SEQUENCE {
69:           OBJECT IDENTIFIER countryName (2 5 4 6)
74:             PrintableString 'US'
: }

Internet-Draft            SIP Secure Call Flows                 May 2006

:                  }
78:              SET {
80:                SEQUENCE {
82:                  OBJECT IDENTIFIER stateOrProvinceName(2 5 4 8)
87:                  PrintableString 'California'
:                  }
:                }
99:              SET {
101:                SEQUENCE {
103:                  OBJECT IDENTIFIER localityName (2 5 4 7)
108:                  PrintableString 'San Jose'
:                }
:              }
118:              SET {
120:                SEQUENCE {
122:                  OBJECT IDENTIFIER organizationName (2 5 4 10)
127:                  PrintableString 'sipit'
:                }
:              }
134:              SET {
136:                SEQUENCE {
138:                  OBJECT IDENTIFIER
:                    organizationalUnitName (2 5 4 11)
143:                  PrintableString 'Sipit Test Certificate Authority'
:                }
:              }
177:            INTEGER 01 95 00 71 02 33 00 58
:            }
187:          SEQUENCE {
189:            OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
:          }
196:          SEQUENCE {
198:            OBJECT IDENTIFIER rsaEncryption(1 2 840 113549 1 1 1)
209:            NULL
:          }
211:          OCTET STRING
:            16 85 D7 B8 08 C6 32 D5 85 7D 26 0F F8 89 DA D0
:            B8 FE 96 FB 40 C9 0E 52 C7 FE A5 87 55 F7 1A 86
:            29 80 CC B0 75 A3 72 DD 76 80 6B 2C 8B C0 14 EA
:            49 FE 18 8F A6 27 BC 5B 60 C1 FE 15 4D 2A 42 DD
:            33 F8 0D D0 77 11 73 82 31 4D 31 66 B1 CF 95 F0
:            9D EE DF 81 E3 54 DF 8C 7B 63 70 D4 93 B5 AE E0
:            D4 90 DB BE D8 0B 3B C2 99 6A FE 5A F0 E9 F0 DF
:            85 F2 A6 8C 28 33 0D 77 04 59 78 06 E5 0E 48 78
:          }
9. Test Consideration

This section describes some common interoperability problems. Implementers should verify that their clients do the correct things and perhaps make their clients forgiving in what they receive, or at least have them produce reasonable error messages when interacting with software that has these problems.

A common problem is that some SIP clients incorrectly only do SSLv3 and do not support TLS.

Many SIP clients were found to accept expired certificates with no warning or error.

TLS and S/MIME can provide the identity of the peer that a client is communicating with in the Subject Alternative Name in the certificate. The software must check that this name corresponds to the identity the server is trying to contact. If a client is trying to set up a TLS connection to good.example.com and it gets a TLS connection set up with a server that presents a valid certificate but with the name evil.example.com, it must generate an error or warning of some type. Similarly with S/MIME, if a user is trying to communicate with sip:fluffy@example.com, one of the items in the Subject Alternate Name set in the certificate must match.

Some implementations used binary MIME encodings while others used base64. The preferred form is binary.

In several places in this draft, the messages contain the encoding for the SHA-1 digest algorithm identifier. The preferred form for encoding as set out in Section 2 of RFC 3370 [10] is the form in which the optional AlgorithmIdentifier parameter field is omitted. However, RFC 3370 also says the receives need to be able to receive the form in which the AlgorithmIdentifier parameter field is present and set to NULL. Examples of the form using NULL can be found in Section 4.2 of RFC 4134 [11]. Receivers really do need to be able to receive the form that includes the NULL because the NULL form, while not preferred, is what was observed as being generated by most implementations. Implementers should also note that if the algorithm is MD5 instead of SHA1, then the form that omits the AlgorithmIdentifier parameters field is not allowed and the sender has to use the form where the NULL is included.
The preferred encryption algorithm for S/MIME in SIP is AES as defined in RFC 3853 [9].

Interoperability was generally better when UAs did not attach the senders' certificates. Attaching the certificates significantly increases the size of the messages, and since it cannot be relied on, it does not turn out to be useful in most situations.

10. IANA Considerations

No IANA actions are required.

11. Acknowledgments

Many thanks to the developers of all the open source software used to create these call flows. This includes the underlying crypto and TLS software used from openssl.org, the SIP stack from www.resiprocate.org, and the SIMPLE IMPP agent from www.sipimp.org. The TLS flow dumps were done with SSLLDump from http://www.rtfm.com/ssldump. The book "SSL and TLS" [13] was a huge help in developing the code for these flows. It’s sad there is no second edition.

Thanks to Jim Schaad, Russ Housley, Eric Rescorla, Dan Wing, Tat Chan, and Lyndsay Campbell who all helped find and correct mistakes in this document.

12. References

12.1. Normative References


12.2. Informative References


Appendix A. Making Test Certificates

These scripts allow you to make certificates for test purposes. The certificates will all share a common CA root so that everyone running these scripts can have interoperable certificates. WARNING - these certificates are totally insecure and are for test purposes only. All the CA created by this script share the same private key to facilitate interoperability testing, but this totally breaks the security since the private key of the CA is well known.

The instructions assume a Unix-like environment with openssl installed, but openssl does work in Windows too. Make sure you have openssl installed by trying to run "openssl". Run the makeCA script found in Appendix A.1; this creates a subdirectory called demoCA. If
the makeCA script cannot find where your openssl is installed you will have to set an environment variable called OPENSSLDIR to whatever directory contains the file openssl.cnf. You can find this with a "locate openssl.cnf". You are now ready to make certificates.

To create certs for use with TLS, run the makeCert script found in Appendix A.2 with the fully qualified domain name of the proxy you are making the certificate for. For example, "makeCert host.example.net". This will generate a private key and a certificate. The private key will be left in a file named domain_key_example.net.pem in pem format. The certificate will be in domain_cert_example.net.pem. Some programs expect both the certificate and private key combined together in a PKCS12 format file. This is created by the script and left in a file named example.net.p12. Some programs expect this file to have a .pfx extension instead of .p12 - just rename the file if needed. A file with a certificate signing request, called example.net.csr, is also created and can be used to get the certificate signed by another CA.

A second argument indicating the number of days for which the certificate should be valid can be passed to the makeCert script. It is possible to make an expired certificate using the command "makeCert host.example.net 0".

Anywhere that a password is used to protect a certificate, the password is set to the string "password".

The root certificate for the CA is in the file root_cert_fluffyCA.pem.

For things that need DER format certificates, a certificate can be converted from PEM to DER with "openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER".

Some programs expect certificates in PKCS#7 format (with a file extension of .p7c). You can convert these from PEM format to PKCS#7 with "openssl crl2pkcs7 -nocrl -certfile cert.pem -certfile demoCA/cacert.pem -outform DER -out cert.p7c"

IE, Outlook, and Netscape can import and export .p12 files and .p7c files. You can convert a pkcs7 certificate to PEM format with "openssl pkcs7 -in cert.p7c -inform DER -outform PEM -out cert.pem".

The private key can be converted to pkcs8 format with "openssl pkcs8 -in a_key.pem -topk8 -outform DER -out a_key.p8c"

In general, a TLS client will just need the root certificate of the CA. A TLS server will need its private key and its certificate.
These could be in two PEM files or one .p12 file. An S/MIME program will need its private key and certificate, the root certificate of the CA, and the certificate for every other user it communicates with.

A.1. makeCA script

#!/bin/sh
#set -x
rm -rf demoCA
mkdir demoCA
cd demoCA/certs
cd demoCA/crl
cd demoCA/newcerts
cd demoCA/private
echo "01" > demoCA/serial
hexdump -n 4 -e '4/1 "%04u"' /dev/random > demoCA/serial
touch demoCA/index.txt

# You may need to modify this for where your default file is
# you can find where yours in by typing "openssl ca"
for D in /etc/ssl /usr/local/ssl /sw/etc/ssl /sw/share/ssl; do
  CONF=${OPENSSLDIR:=$D}/openssl.cnf
  [ -f $CONF ] && break
done
if [ ! -f $CONF ]; then
  echo "Can not find file $CONF - set your OPENSSLDIR variable"
  exit
fi
cp $CONF openssl.cnf
cat >> openssl.cnf <<EOF
[cj_cert]
subjectAltName=\$(ENV::ALTNAME)
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
#authorityKeyIdentifier=keyid,issuer:always

[cj_req]
basicConstraints = CA:FALSE
subjectAltName=\$(ENV::ALTNAME)
subjectKeyIdentifier=hash
#authorityKeyIdentifier=keyid,issuer:always
#keyUsage = nonRepudiation, digitalSignature, keyEncipherment

EOF
EOF
cat > demoCA/private/cakey.pem <<EOF
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,4B47A07A3DE342E

aHmlPa+Zr0V6v+Jk0SclxpxoG3j0zu0y0Vf9rzq2bzkVBLU6xhWwJMDqwa8dH3fCRLhMGIUVnynmYXth9w1lqPXMQBHJcKvpV/Smgf/FbYk98Smo2izH0n1iIuNOu2zrubMiaBhp0AZ/OCTvxU00BdKSN1R39UCOdngkEQzp9Vbw7173y5H9GMHJPtGLFy3RhS3TvLFLA2JZjms/wZ/9QM8GjyJrIhmGRJRJeI2Gv4Yr1u6YH1fJxtXmeds8L3h85vJyAjnk9j9tJasA/BeCzd7p7fZbgodLdWu1l+kkpff5h1uHz7aoaum0M1EXBE4tcDHsfglqEsDMiI/U/+rWfk1PrzYlk1w3p8S03vulkDm1ftT76w7d
MrBg4+crHA6qYn6EPWB37OBtfEqA1InIc1idWzo9a0BTP4E4OJ0A0PcZ/2JgT
PaKySg0ooHQ8AHNQebech6M5LFEExpaOAdJKrquaKcc2HeUxAIYpac5/7dr13i1oUloqUnM1Ga3eL7P7ZImsZKCFH28owuU4g66mmJath2ODDy3mfh6LyaimD7v4
SAI1krEHFXSyovrTjyMfSftqyXUxurar2Dqaxs6o7/eG1lRxlifGMLY9ceu/Y/8FNM
LE7p+ca19H5hHHzx1ilieaK43u/XvbXH15mql/fZdkUJBjsjBvX0HR8eQ12C9
YJDMOPLAdewcIyOyKAOAY59oN9d41of7y2tN9KwNdslROYH7mN1jgQMenhXCLN+Ns
vVU5/7/ug2FhZqfS46clWdmsvupqDp7TbtMeaH/PXjySbr0izff0xOq==
-----END RSA PRIVATE KEY-----
EOF
cat > demoCA/cacert.pem <<EOF
-----BEGIN CERTIFICATE-----
MIIDJDCAo2gAwIBAgIBADANBgkqhkiG9w0BAQUFBwEBMwswCQYDVQQGEwJVUzETMBEGA1UECBIKMQoFMSAwHAgIBADANBgkqhkiG9w0BAQUFBwEBMDgE

# uncomment the following lines to generate your own key pair
#openssl req -newkey rsa:1024 -passin pass:password \
#    -passout pass:password \
#    -sha1 -x509 -keyout demoCA/private/cakey.pem \
#    -out demoCA/cacert.pem -days 3650 <<EOF
#US
#California
#San Jose
#sipit
#Sipit Test Certificate Authority
#
#EOF

openssl crl2pkcs7 -nocrl -certfile demoCA/cacert.pem \
    -outform DER -out demoCA/cacert.p7c

cp demoCA/cacert.pem root_cert_fluffyCA.pem

A.2. makeCert script

#!/bin/sh
#set -x

if [ $# == 1 ]; then
  DAYS=1095
elif [ $# == 2 ]; then
  DAYS=$2
else
  echo "Usage: makeCert test.example.org [days]"
  echo "   makeCert alice@example.org [days]"
  echo "days is how long the certificate is valid"
  echo "days set to 0 generates an invalid certificate"
  exit 0
fi

ADDRT=$1

echo "making cert for ${ADDRT}"

rn -f ${ADDRT}_*.pem
rn -f ${ADDRT}.p12

case ${ADDRT} in
  *:*)
    ALTNAME="URI:${ADDRT}" ;
  ;;
  *@*)
  ;;
  *)
    ALTNAME="DNS:${ADDRT}" ;
  esac
rm -f demoCA/index.txt
touch demoCA/index.txt
rm -f demoCA/newcerts/*

export ALTBINAME

openssl genrsa -out ${ADDR}_key.pem 1024
openssl req -new -config openssl.cnf -reqexts cj_req -sha1 -key ${ADDR}_key.pem -out ${ADDR}.csr -days ${DAYS} <<EOF
US
California
San Jose
sipit
${ADDR}
EOF

if [ $DAYS == 0 ]; then
    openssl ca -extensions cj_cert -config openssl.cnf -passin pass:password -policy policy_anything -md sha1 -batch -notext -out ${ADDR}_cert.pem -startdate 990101000000Z -enddate 000101000000Z -infiles ${ADDR}.csr
else
    openssl ca -extensions cj_cert -config openssl.cnf -passin pass:password -policy policy_anything -md sha1 -days ${DAYS} -batch -notext -out ${ADDR}_cert.pem -infiles ${ADDR}.csr
fi

openssl pkcs12 -passin pass:password -passout pass:password -export -in ${ADDR}_cert.pem -inkey ${ADDR}_key.pem -name ${ADDR} -certfile demoCA/cacert.pem

openssl x509 -in ${ADDR}_cert.pem -noout -text

case ${ADDR} in
  *@*) mv ${ADDR}_key.pem user_key_${ADDR}.pem;  
             mv ${ADDR}_cert.pem user_cert_${ADDR}.pem ;;
  *) mv ${ADDR}_key.pem domain_key_${ADDR}.pem;  
               mv ${ADDR}_cert.pem domain_cert_${ADDR}.pem ;;
esac
Appendix B. Certificates for Testing

This section contains various certificates used for testing in PEM format.

Fluffy’s certificate.

-----BEGIN CERTIFICATE-----
MIICzjCCajegAwIBAgIIAZUAcQIzAFgwDQYJKoZIhvcNAQEFBQAwDELMb290b2NvQmF0ZURpZSBD
xN0MjAwHhcNMTg0NTMwNzExMDAwWhcNMDgwMjAzMTg0OMQwWjBjMQswCQYD
VQQGQXwvJwUETMBAQAAMQswCQYD
DjAmIgNBQBAOToqgclKMQswCQYD
U2F1e3MyMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEwMCow
BjAmIgNBQBAOToqgclKMQswCQYD
U2F1e3MyMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEwMCow
-----END CERTIFICATE-----

Fluffy’s private key

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDKq5ubTjzVRTz0AKy2qLnso0bliuZvokKq6hiLFzzM+T21iWiM9
VR8FBT52mAE61vr1N1HUVy77v1jGvK18jmuQv5Whj1Z0Sp91cYRZwC5ogn
ahesyiqE5mLY8em1Gj1+Dg+QpaZ5IbgwZ77Dav2Cmqs4fAlKKNWPN10NRmzIDAQAB
AoGaGxqtxw0jXiB716/PCs+stUt+UiRwxIT30fddHONACRr8RmgqM1khAzf7XMo1
kegJjmrF3+K614g41OcnL3y1wVctzJf12QTuVzAsvazq2z14+pbN4aLaB+jPN+q
Brq5XADYxV7Hftuakze2psgnHYw+t3HwAoVgKjLDKKHdrrOEQCDP+G+GsJdmnrip
wZ9F0xKQkHqB07ZONHdxuS5Auk1hGnFh1udqC7EscaCqLN82RS4gn/WdFnbB
WBDRavxAvbKEAe3o6nNMMOMKdsuqBbGyFpavDPVmw973wtEchI6MgwY5OUHdKAd
hs09yVGrQopjnomHIEo51UhPxlJ1+Xxv1wJBAL0Yr1y4DFx6U/4WbqB1i6PwL62gk
q759AH9nt8Et0x6SNbneq0Y+Njym/2Nc3J0eE2BQezi6gsRcp6JLdudhECDQP+1
V7EwhCUNv8cbkJXWmcbmy6Cpwyx/XF2K7DRBTVTXC6WXeRo17jIzlfjY8
nNzWP9I4A4m6h3skRrkCQA+1er++Tw4xypEiij70K0bfjJULrhCMN9VWx6DrzekO
32pulB7yuuxrbcmM21JKQ1HL0sw7egscepX+S6ly8=
-----END RSA PRIVATE KEY-----

Kumiko’s certificate

-----BEGIN CERTIFICATE-----
MIICzjCCAjegAwIBAgIIZAUAcQIZAFcwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVXMVzEAkARBgNVBAtgCmNhbiBkbn3IuMQswgZTAwHnDigzCEMBEEGBAQG
MR0GswgZTAwHnDiMEkgCgYGBQMEHwYIKoZIhvcNAQELBQAwgZIwDQYJKoZIhvc
NAQEFBQAwcDELMAkGA1UEBhMCVXMVMRkwGgYGBQMEHwYIKoZIhvcNAQELBwIE
MRAwIGA1UEBxMgU2FuIEpvc2Ux
-----END CERTIFICATE-----

Kumiko’s private key

-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDV+nYToVLni/tn5csp3rkg8By2ajKp7tyEoiI4v12RKFVARSB2
I7cw5oSIuNhJeHHkK/vSOoGoJ3uIay/NwK9WkcY3lAe+J1u2u9edwvMQOpwl
ZykJDkZ8w3gRRH5/gqskT1sk0ihB91XFLn5w2nnmmw6aFmXADwDwqQIDMQAB
AoGBANJktWrxyanxc471I/dPewVjVqoHea7jq8yS6oyr13cDPvnpWu1mFkCNumXV
W/mNum89H23V8dN03gcsIdxpcnuwmm7TiHsQXjBts0n0DBoww8T+sWtS
9njR4m1+860f6I+eTVa3d2weCE/pEOKhtFzVxixosYmncAAbAkE6aqKEwRlZI/V
u2z2B1euafkJQdFpBooaQh+9qy5uWgZz6Xj6tM8YUsVqR/NCg8auroRC5uWD
yonN92kqPQJBAv0/jP9yy0N2CVs4MP5SQD01RA0uruMz6VlnURQ0/8uBmhVeIc
nkvq9xhxJH7mmusEY3IzvRkm4FV4RZcyBEQCHHtI5/Tq+Mm2tKmLXkkfYf+MOoP
y2wuiG0at2Ls82VjVbVhnJ55Fbvdw6+721qFvZ2eXv3+wgI9orOGoDXnwECQDrE
I5BPsczGGhkUBhKhp+4kSTkW89jHJyVpDAKEOHkKHHecZAh0Hv9sWwHg961091JD
BcANjtLhmHz9fereRpwBEQCCqCuIn02CMXf5y5j4n1mCRQw6KWBWjY68aN4Qj/g3
SV+1HmtCmC05k3e/V6vG0Kn+MV3c+14JGdsRM+9HqcZ
-----END RSA PRIVATE KEY-----

Certificate for example.com
-----BEGIN CERTIFICATE-----
MIICjDCCAfWgAwIBAgIITAUAcQIzAFyDQYJKoZIhvcNAQEFBQAwgAxCzAJBgNV
BAgTCFNhbiBKb3NlMQ4wDAYDVQQKEwJVUzETMBEGA1UECBMIU2FuIEpvc2Ux
DjAMBgNVBAoTBXNpcGxhYWQwEQQDQgU76tmOYj47Br5Hw/sb+73CJtPf/ta
-----END CERTIFICATE-----

Private key for example.net
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDbDgj9vP+/HNWyRQQxNoRi4n/j+Rk2bqOyIyiClDwDwAvd
-----BEGIN RSA PRIVATE KEY-----

Appendix C  Message Dumps

This section contains a base64 encoded gzipped, compressed tar file of
various CMS messages used in this document. Saving the data in a file
foo.tgz.b64 then running a command like "openssl base64 -d -in
foo.tgz.b64 | tar txfz -" would recover the CMS messages and allow
them to be used as test vectors.

H4sIAO0R...
Authors’ Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421 9990
Email: fluffy@cisco.com

Kumiko Ono
NTT Corporation
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 4508
Email: ono.kumiko@lab.ntt.co.jp

Robert Sparks (editor)
Estacado Systems

Email: RjS@estacado.net
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.