Network Working Group
Internet-Draft
Expires: December 27, 2006

C. Jennings
Cisco Systems
K. Ono
NTT Corporation
R. Sparks, Ed.
Estacado Systems
June 25, 2006

Example call flows using Session Initiation Protocol (SIP) security mechanisms
draft-ietf-sip-sec-flows-01

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. Introduction

This document is informational and is not normative on any aspect of SIP.

SIP with TLS[4] implementations are becoming very common. Several implementations of the S/MIME[7] portion of SIP[2] are also becoming available. After several interoperability events, it is 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 5. The certificates for the hosts used are shown in Section 4.2, and the CA certificates used to sign these are shown in Section 4.1.

The text from Section 6.1 through Section 6.3 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 4.3. These host certificates are signed with the same CA certificate.

Section 7 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. The certificates used while creating the examples and test messages in this document are made available in Appendix B.

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

2. 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].

3. 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 needed to check Certificate Revocation Lists (see [3]) as that is not part of the SIP call flow.

4. Certificates

4.1. 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 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:
               4b:9d:fa:72:78:c6:b0:95:18:a7:e0:8c:79:ba:5d:
               a4:ae:1e:21:2d:9d:f1:0b:1c:cf:bd:5b:29:b3:90:
               13:73:66:92:6e:df:4c:b3:b3:1c:1f:2a:82:0a:ba:
               07:4d:52:b0:f8:37:7b:e2:0a:27:30:70:dd:f9:2e:
               03:ff:2a:76:cd:df:87:1a:bd:71:eb:e1:99:6a:c4:
               7f:8e:74:a0:77:85:04:e9:41:ad:fc:03:b6:17:75:
               aa:33:ea:0a:16:d9:fb:79:32:2e:f8:cf:4d:c6:34:
               a3:ff:1b:d0:68:28:e1:9d:e5
           Exponent: 65537 (0x10001)
   X509v3 extensions:
     X509v3 Subject Key Identifier:
       6B:46:17:14:EA:94:76:25:80:54:6E:13:54:DA:A1:E3:54:14:A1:B6
     X509v3 Authority Key Identifier:
       6B:46:17:14:EA:94:76:25:80:54:6E:13:54:DA:A1:E3:54:14:A1:B6
       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
    96:6d:1b:ef:d5:91:93:45:7c:5b:1f:cf:c4:aa:47:52:0b:34:
    a8:50:fa:ec:fa:b4:2a:47:4c:5d:41:a7:3d:c0:d6:3f:9e:56:
    5b:91:1d:ce:a8:07:b3:1b:a4:9f:9a:49:6f:7f:e0:ce:83:94:
    71:42:af:fe:63:a2:34:dc:b4:5e:a5:ce:ca:79:50:e9:6a:99:
    4c:14:69:e9:7c:ab:22:6c:44:cc:8a:9c:33:6b:23:50:42:05:
    1f:e1:c2:81:88:5f:ba:e5:47:bb:85:9b:83:25:ad:84:32:ff:
    2a:5b:8b:70:12:11:83:61:c9:69:15:4f:58:a3:3c:92:d4:e8:
    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=  41 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.]...!-......
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{..'0p.
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 72 30 70 31   .T...T....t.r0p1
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....Californi
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 0a 13 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 20 43 65 72 74 69 66   ipit Test Certif
69 63 61 74 65 20 41 75-74 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
  654:l=   5 prim:      OCTET STRING
30 03 01 01 ff                                    0....
  661:l=  13 cons:  SEQUENCE
  663:l=   9 prim:   OBJECT            :sha1WithRSAEncryption
```

```
  674:l=   0 prim:   NULL
  676:l= 129 prim:  BIT STRING
00 96 6d 1b ef d5 91 93-45 7c 5b 1f cf c4 aa 47   ..m.....E|[....G
52 0b 34 a8 50 fa ec fa-b4 2a 47 4c 5d 41 a7 3d   R.4.P....*GL]A.=
c0 d6 3f 9e 56 5b 91 1d-ce a8 07 b3 1b a4 9f 9a   ..?.V[..........
49 6f 7f e0 ce 83 94 71-42 af fe 63 a2 34 dc b4   Io.....qB..c.4..
5e a5 ce ca 79 50 e9 6a-99 4c 14 69 e9 7c ab 22   ^...yP.j.L.i.|."
6c 44 cc 8a 9c 33 6b 23-50 42 05 1f e1 c2 81 88   lD...3k#PB......
5f ba e5 47 bb 85 9b 83-25 ad 84 32 ff 2a 5b 8b   _..G....%..2.*[.
70 12 11 83 61 c9 69 15-4f 58 a3 3c 92 d4 e8 6f   p...a.i.OX.<...o
52                                                R
```

4.2. 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):
                   00:e6:31:76:b5:27:cc:8d:32:85:56:70:f7:c2:33:
                   33:32:26:42:5e:3c:68:71:7b:1f:79:50:d0:72:27:
                   3b:4a:af:f2:ce:d1:0c:bc:c0:5f:31:6a:43:e7:7c:
                   ad:64:bd:c7:e6:25:9f:aa:cd:2d:90:aa:68:84:62:
                   7b:05:be:43:a5:af:bb:ea:9d:a9:5b:a4:53:9d:22:
                   8b:da:96:2e:1f:3f:92:46:b8:cc:c8:24:3c:46:cd:
                   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:
                   b3:bb:72:7a:98:e5:ff:9f:81
               Exponent: 65537 (0x10001)
       X509v3 extensions:
           X509v3 Subject Alternative Name:
               DNS:example.com
           X509v3 Basic Constraints:
               CA:FALSE
           X509v3 Subject Key Identifier:
               22:EA:CB:38:66:1D:F1:96:0C:9A:47:B6:BB:1C:52:
               44:B0:77:65:8D
   Signature Algorithm: sha1WithRSAEncryption
       ae:eb:49:ed:1e:f1:8d:26:a9:6d:03:82:92:d5:df:44:c4:1e:
       1f:07:75:88:37:e4:76:97:35:12:59:98:79:78:16:6e:3b:b1:
       c0:2b:db:85:02:6b:74:c9:5b:19:92:da:7e:f5:41:0b:bc:d2:
       dd:45:aa:6f:be:24:dc:48:57:66:d9:2e:82:df:9e:8d:70:03:
       73:75:ef:8f:7a:56:4c:cc:42:bd:31:45:b0:5e:ff:d1:3b:c4:
       82:ee:fd:a7:c1:10:34:eb:81:49:1a:6b:86:7e:c7:61:1d:b3:
       b9:0a:02:bd:84:f8:47:af:cf:f1:a8:73:a8:31:1d:20:7a:06:
       7f:ac
```

4.3. 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:
                   b9:ec:d2:76:e2:b9:9b:e8:28:aa:ba:86:22:c5:cf:
                   33:3e:4f:6d:56:21:ae:bd:54:84:7c:14:14:f9:7d:
                   99:85:00:4e:93:d6:fd:6b:d4:d1:d4:55:8e:c9:89:
                   b1:af:2b:5f:23:99:4a:95:e5:68:65:64:1d:12:a7:
                   db:d3:d5:97:18:47:35:9c:e6:88:27:9d:a8:6c:ca:
                   2a:84:e6:62:d8:f1:e9:a2:1a:39:7e:0e:0f:90:a5:
                   a6:79:21:bc:2a:67:b4:dd:69:90:82:9a:ae:1f:02:
                   52:8a:58:d3:f5:d0:d4:66:67
               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:
               EC:DA:98:5E:E9:F7:F7:D7:EC:2B:29:4B:DA:25:EE:C7:C7:
               7E:95:70
   Signature Algorithm: sha1WithRSAEncryption
       4c:46:49:6e:01:48:e2:d4:6e:d7:48:a1:f3:7b:c8:a5:98:37:
       a5:44:46:58:9f:4a:37:7d:90:fb:5f:ff:36:bd:67:31:f0:29:
       de:0a:e2:ea:b9:f0:5c:9f:ad:a0:de:e5:4e:42:8f:11:d8:41:
       ea:68:be:db:c2:1e:fa:e5:8a:2d:7f:66:13:29:e9:da:8f:fb:
       80:bf:7e:5e:b6:04:ad:08:5e:58:95:b7:c5:38:85:d5:65:31:
       ad:80:cb:28:a7:4c:ad:11:fd:41:3b:37:77:5a:de:85:96:3d:
       66:eb:5f:9a:f8:60:5f:8e:b1:fc:4a:43:53:b6:11:4d:2e:f4:
       3d:ff
```

5. Callflow with Message Over TLS

5.1. 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 [16].

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
```

```
            ea c5 aa de 59 41 e4 f3 9e 1c 1c a9 48 f5 0a 3f
            5e c3 50 23 15 d7 46 1d 69 79 76 ba 5e c8 ac 39
            23 71 d0 0c 18 a6 a9 77 0f 7d 49 61 ef 6f 8d 32
            54 f5 a4 1d 19 33 0a 64 ee 56 91 9b f4 f7 50 b1
            11 4b 81 46 4c 36 df 70 98 04 dc 5c 8a 16 a9 2e
            58 67 ae 5e 7a a9 44 2b 0b 7c 9c 2f 16 25 1a e9
   1 6  0.0183 (0.0000)  C>SV3.1(1)  ChangeCipherSpec
   1 7  0.0183 (0.0000)  C>SV3.1(48)  Handshake
   1 8  0.0630 (0.0447)  S>CV3.1(1)  ChangeCipherSpec
   1 9  0.0630 (0.0000)  S>CV3.1(48)  Handshake
   1 10 0.3274 (0.2643)  C>SV3.1(32)  application_data
   1 11 0.3274 (0.0000)  C>SV3.1(720)  application_data
   1 12 0.3324 (0.0050)  S>CV3.1(32)  application_data
   1 13 0.3324 (0.0000)  S>CV3.1(384)  application_data
   1    9.2491 (8.9166)  C>S  TCP FIN
   1    9.4023 (0.1531)  S>C  TCP FIN
```

5.2. MESSAGE Message Over TLS

Once the TLS session is set up, the following MESSAGE message (as defined in [13] 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, the <allOneLine>convention from [11] is used to break long lines. The actual message does not contain the linebreaks contained within those tags.

```
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
```

6. Callflow with S/MIME-secured Message

6.1. 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.

```
MESSAGE sip:kumiko@example.net SIP/2.0
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=0c523b42
Via: SIP/2.0/UDP 68.122.119.3:5060;
     branch=z9hG4bK-d87543-16a1192b7960f635-1--d87543-;rport
Call-ID: 27bb7608596d8914@Y2ouY2lzY28uc2lwaXQubmV0
CSeq: 1 MESSAGE
Contact: <sip:fluffy@68.122.119.3:5060>
Max-Forwards: 70
Content-Transfer-Encoding: binary
Content-Type: multipart/signed;boundary=151aa2144df0f6bd;\
              micalg=sha1;protocol="application/pkcs7-signature"
Date: Sat, 19 Nov 2005 23:34:50 GMT
User-Agent: SIPimp.org/0.2.5 (curses)
Content-Length: 639

--151aa2144df0f6bd
Content-Type: text/plain
Content-Transfer-Encoding: binary

hello
--151aa2144df0f6bd
Content-Type: application/pkcs7-mime;name=smime.p7s
Content-Disposition: attachment;handling=required;filename=smime.p7s
Content-Transfer-Encoding: binary

******************
* BINARY BLOB 1   *
******************
--151aa2144df0f6bd--
```

It is important to note that the signature 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.

```
Content-Type: text/plain
Content-Transfer-Encoding: binary

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].

```
  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'
    :                  }
    :                }
 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
    :            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
    :          }
    :        }
    :      }
    :    }
    :  }
```

6.2. MESSAGE Message with Encrypted Body

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

```
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: 031be67669ea9799@Y2ouY2lzY28uc2lwaXQubmV0
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 {
```

```
 77:               SEQUENCE {
 79:                 OBJECT IDENTIFIER localityName (2 5 4 7)
 84:                 PrintableString 'San Jose'
   :                 }
   :               }
 94:             SET {
 96:               SEQUENCE {
 98:                 OBJECT IDENTIFIER organizationName (2 5 4 10)
103:                 PrintableString 'sipit'
   :                 }
   :               }
110:             SET {
112:               SEQUENCE {
114:                 OBJECT IDENTIFIER
   :                   organizationalUnitName (2 5 4 11)
119:                 PrintableString
                      'Sipit Test Certificate Authority'
   :                 }
   :               }
   :             }
153:           INTEGER 01 95 00 71 02 33 00 57
   :           }
163:         SEQUENCE {
165:          OBJECT IDENTIFIER rsaEncryption(1 2 840 113549 1 1 1)
176:          NULL
   :          }
178:         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
   :         }
   :       }
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
   :           50 9E 44 AA A5 54 C3 5C 0D 9A DF 65 F7 47 36 99
   :         }
353:       [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
```

```
:          61 24 E1 62 B0 DA 03 53 78 F8 B7 CD B2 11 68 57
:          BE 5F 13 49 B9 5E AB 6F 6E 26 2D 8A A5 9E E5 10
:        }
:       }
:    }
:  }
```

6.3. 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@Y2ouY2lzY28uc2lwaXQubmV0
   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
   |--See note about stuff in this box --------------------|
   |Content-Type: application/pkcs7-mime;                  |
   |              smime-type=enveloped-data;name=smime.p7m |
   |-------------------------------------------------------|
   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'
   :                  }
   :                }
 75:              SET {
 77:                SEQUENCE {
 79:                  OBJECT IDENTIFIER localityName (2 5 4 7)
 84:                  PrintableString 'San Jose'
   :                  }
   :                }
 94:              SET {
 96:                SEQUENCE {
 98:                  OBJECT IDENTIFIER organizationName (2 5 4 10)
103:                  PrintableString 'sipit'
   :                  }
   :                }
110:              SET {
112:                SEQUENCE {
114:                  OBJECT IDENTIFIER
   :                    organizationalUnitName (2 5 4 11)
119:                  PrintableString
                        'Sipit Test Certificate Authority'
   :                  }
   :                }
   :              }
153:            INTEGER 01 95 00 71 02 33 00 57
   :            }
163:          SEQUENCE {
165:           OBJECT IDENTIFIER rsaEncryption(1 2 840 113549 1 1 1)
176:           NULL
   :           }
178:          OCTET STRING
   :            69 B3 A3 61 F4 F8 63 4F 46 0A 1A AB 0F 1B 16 09
   :            DB 3A A9 12 3B 23 F0 C9 4E 68 04 15 AB 42 4F 66
   :            FA EF 8D C4 86 88 41 BA 53 A3 88 49 54 E3 0E EB
   :            E3 69 63 5A DF 77 2A 8A 1E 42 7E E4 A7 DB CF 90
   :            7E 90 47 FD 20 C9 B2 3B 2F A5 42 2A 68 66 9A 25
   :            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
    :          }
  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 98 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 94 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'
    :                  }
```

```
   :              }
 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
   :        }
   :      }
```

```
:      }
:    }
:  }
```

7. Observed Interoperability Issues

This section describes some common interoperability problems. Implementers should verify that their clients do the correct things and when possible make their clients forgiving in what they receive. Implementations should take extra care to produce reasonable error messages when interacting with software that has these problems.

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.

When used with SIP, TLS and S/MIME 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. Implementations should send only binary but must be prepared to receive either.

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 recipients 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 [12]. 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].

Observed interoperability has been better when UAs did not attach the senders' certificates. Attaching the certificates significantly increases the size of the messages, and since it can not be relied on, it does not turn out to be useful in most situations.

8. Additional test scenarios

This section provides the beginning of a list of tests that implementations should perform while developing systems that use S/MIME and TLS for SIP.

Much of the required behavior for inspecting certificates when using S/MIME and TLS with SIP is currently underspecified. The non-normative recommendations in this document capture the current folklore around that required behavior, guided by related normative works such as [14] (particulary the section on Domain Names and Subordination) and [15] section 3.1. To summarize that non-normative lore:

- For S/MIME the peer's URI must appear in the subjectAltName of the peer's certifcate as a uniformResourceIdentifier field.
- For TLS the peer's hostname (as fed into 3263 for resolution for locating the peer) must appear as
  - an exact match in a dNSName entry in the subjectAltName if there are any dNSNames in the subjectAltName. (Wildcard matching is not allowed against these dNSName entries)
  - the most specific CommonName in the Subject field if there are no dNSName entries in the subjectAltName at all (which is not the same as there being no matching dNSName entries). This match can be either exact, or against an entry that uses the wildcard matching character '*'

For each of these tests, an implementation will proceed past the verification point only if the certificate is "good". S/MIME protected requests presenting bad certificate data will be rejected. S/MIME protected responses presenting bad certificate information will be ignored. TLS connections involving bad certificate data will not be completed.

1. S/MIME : Good peer certificate
2. S/MIME : Bad peer certificate (peer URI does not appear in subjAltName)
3. S/MIME : Bad peer certificate (valid authority chain does not end at a trusted CA)

4. S/MIME : Bad peer certificate (the current time does not fall within the period of validity)
5. S/MIME : Bad peer certificate (certificate or cert in authority chain has been revoked)
6. TLS : Good peer certificate (hostname appears in dNSName in subjAltName)
7. TLS : Good peer certificate (no dNSNames in subjAltName, hostname appears in CN of Subject)
8. TLS : Bad peer certificate (no match in dNSNames or in the Subject CN)
9. TLS : Bad peer certificate (valid authority chain does not end at a trusted CA)
10. TLS : Bad peer certificate (the current time does not fall within the period of validity)
11. TLS : Bad peer certificate (certificate or cert in authority chain has been revoked)

9. IANA Considerations

No IANA actions are required.

10. 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 SSLDump from http://www.rtfm.com/ssldump. The book "SSL and TLS" [16] 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.

Vijay Gurbani and Alan Jeffrey contributed much of the additional test scenario content.

11. Changelog

(RFC Editor: remove this section)

-00 to -01

* Incorporated the Test cases from Vijay Gurbani's and Alan Jeffrey's Use of TLS in SIP draft
* Began to capture the folklore around where identities are carried in certificates for use with SIP
* Removed the message dump archive pending verification (will return in -02)

12. Known Problems with this version

The flows, encryption and signatures captured in this document were manually collected from actual test runs. Those runs need to be reproduced so the source messages can be modified (as below for instance). This reproduction is being automated, but is not yet complete.

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

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

13. References

13.1. Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[4] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A., and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[5] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions",

RFC 3546, June 2003.

[6] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

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

[8] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

[9] Peterson, J., "S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)", RFC 3853, July 2004.

[10] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

[11] Sparks, R., Hawrylyshen, A., Johnston, A., Rosenberg, J., and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, May 2006.

13.2. Informative References

[12] Hoffman, P., "Examples of S/MIME Messages", RFC 4134, July 2005.

[13] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[14] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-06 (work in progress), October 2005.

[15] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[16] Rescorla, E., "SSL and TLS - Designing and Building Secure Systems", 2001.

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 filed 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
mkdir demoCA/certs
mkdir demoCA/crl
mkdir demoCA/newcerts
mkdir 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

   cat > demoCA/private/cakey.pem <<EOF
   -----BEGIN RSA PRIVATE KEY-----
   Proc-Type: 4,ENCRYPTED
   DEK-Info: DES-EDE3-CBC,4B47A0A73ADE342E

   aHmlPa+ZrOV6v+Jk0SClxzpxoG3j0ZuyoVkF9rzq2bZkzVBKLU6xhWwjMDqwA8dH
   3fCRLhMGIUVnmymXYhTW9svI1gpFxMBQHJcKpV/SmgFn/fbYk98Smo2izHOniIiu
   NOu2zr+bMiaBphOAZ/OCtVUxUOoBDKN9lR39UCDOgkEQzp9Vbw7l736yu5H9GMHP
   JtGLJyx3RhS3TvLfLAJZhjm/wZ/9QM8GjyJEiDhMQRJVeIZGvv4Yr1u6yYHiHfjX
   tX2eds8Luc83HbSvjAyjnkLtJsAZ/8cFzrd7pjFzbogLdWuil+kpkkf5h1uzh7oa
   um0M1EXBE4tcDHsfg1iqEsDMIei/U+/rWfk1PrzYlklwZp8S03vulkDm1fT76W7d
   mRBg4+CrHA6qYn6EPWB37OBtfEqAfINnIcI1dWzso9A0bTPD4EJO0JA0PcZ/2JgT
   PaKySgooHQ8AHNQebelch6M5LFExpaOADJKrqauKcc2HeUxXaYIpac5/7drIl3io
   UloqUnMlGa3eLP7BZIMsZKCfHZ8oqwU4g6mmmJath2gODRDx3mfhH6yaimDL7v4i
   SAIIkrEHXfSyovrTJymfSfQtYxUraVZDqax6oj/eGllRxliGfMLYG9ceU+yU/8FN
   LE7P+Cs19H5tHHzx1LlieaK43u/XvbXHlB5mqL/fZdkUIBJsjbBVx0HR8eQl2CH9
   YJDMOPLADecwHoyKA0AY59oN9d41oF7yZtN9KwNdslROYH7mNJlqMMenhXCLN+Nz
   vVU5/7/ugZFhZqfS46c1WdmSvuqpDp7TBtMeaH/PXjysBr0iZffOxQ==
   -----END RSA PRIVATE KEY-----
   EOF

   cat > demoCA/cacert.pem <<EOF
   -----BEGIN CERTIFICATE-----
   MIIDJDCCAo2gAwIBAgIBADANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzET
   MBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxDjAMBgNVBAoT
   BXNpcGl0MSkwJwYDVQQLEyBTaXBpdCBUZXN0IENlcnRpZmljYXRlIEF1dGhvcml0
   eTAeFw0wMzA3MTgxMjIxNTJaFw0xMzA3MTUxMjIxNTJaMHAxCzAJBgNVBAYTAlVT
   MRMwEQYDVQQIEwpDYWxpZm9ybmlhMREwDwYDVQQHEwhTYW4gSm9zZTEOMAwGA1UE
   ChMFc2lwaXQxKTAnBgNVBAsTIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUgQXV0aG9y
   aXR5MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDIh6DkcUDLDyK9BEUxkud
   +nJ4xrCVGKfgjHm6XaSuHiEtnfELHM+9WymzkBNzZpJu30yzsxwfKoIKugdNUrD4
   N3viCicwcN35LgP/KnbN34cavXHr4ZlqxH+OdKB3hQTpQa38A7YXdaoz6goW2ft5
   Mi74z03GNKP/G9BoKOGd5QIDAQABo4HNMIHKMB0GA1UdDgQWBBRrRhcU6pR2JYBU
   bhNU2qHjVBShtjCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU2qHjVBShtqF0
   pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcT
   CFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBD
   ZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0B
```

```
AQUFAAOBgQCWbRvv1ZGTRXxbH8/EqkdSCzSoUPrs+rQqR0xdQac9wNY/nlZbkR3O
qAezG6Sfmklvf+DOg5RxQq/+Y6I03LRepc7KeVDpaplMFGnpfKsibETMipwzayNQ
QgUf4cKBiF+65Ue7hZuDJa2EMv8qW4twEhGDYclpFU9YozyS1OhvUg==
-----END CERTIFICATE-----
EOF


# 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

ADDR=$1

echo "making cert for ${ADDR}"
```

```
   rm -f ${ADDR}_*.pem
   rm -f ${ADDR}.p12

   case ${ADDR} in
   *:*) ALTNAME="URI:${ADDR}" ;;
   *@*) ALTNAME="URI:sip:${ADDR},URI:im:${ADDR},URI:pres:${ADDR}" ;;
   *)   ALTNAME="DNS:${ADDR}" ;;
   esac

   rm -f demoCA/index.txt
   touch demoCA/index.txt
   rm -f demoCA/newcerts/*

   export ALTNAME

   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 \
       -out ${ADDR}.p12 -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-----
MIICzjCCAjegAwIBAgIIAZUAcQIzAFgwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwMjAzMTg0OTM0WhcNMDgwMjAzMTg0OTM0WjBiMQswCQYD
VQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2Ux
DjAMBgNVBAoTBXNpcGl0MRswGQYDVQQDFBJmbHVmZnlAZXhhbXBsZS5jb20wgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMqrm5tOPNVFPM4ApjaouezSduK5m+go
qrqGIsXPMz5PbVYhrr1UhHwUFPl9mYUATpPW/WvU0dRVjsmJsa8rXyOZSpXlaGVk
HRKn29PVlxhHNZzmiCedqGzKKoTmYtjx6aIaOX4OD5ClpnkhvCpntN1pkIKarh8C
UopY0/XQ1GZnAgMBAAGjfzB9MFEGA1UdEQRKMEiGFnNpcDpmbHVmZnlAZXhhbXBs
ZS5jb22GFWltOmZsdWZmeUBleGFtcGxlLmNvbYYXcHJlczpmbHVmZnlAZXhhbXBs
ZS5jb20wCQYDVR0TBAIwADAdBgNVHQ4EFgQU7NqYXun399fsKylL2iXux8d+lXAw
DQYJKoZIhvcNAQEFBQADgYEATEZJbgFI4tRu10ih83vIpZg3pURGWJ9KN32Q+1//
Nr1nMfAp3gri6rnwXJ+toN7lTkKPEdhB6mi+28Ie+uWKLX9mEynp2o/7gL9+XrYE
rQheWJW3xTiF1WUxrYDLKKdMrRH9QTs3d1rehZY9ZutfmvhgX46x/EpDU7YRTS70
Pf8=
-----END CERTIFICATE-----
```

Fluffy's private key

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDKq5ubTjzVRTzOAKY2qLns0nbiuZvoKKq6hiLFzzM+T21WIa69
VIR8FBT5fZmFAE6T1v1r1NHUVY7JibGvK18jmUqV5WhlZB0Sp9vT1ZcYRzWc5ogn
nahsyiqE5mLY8emiGjl+Dg+QpaZ5IbwqZ7TdaZCCmq4fAlKKWNP10NRmZwIDAQAB
AoGAXgtxwoh0jBZ716/PcS+sTut+xUiRwxIT30fdHONACRr8RmqM1khAzf7XmMoi
kegJjmrF3+K6l4g4IOcnL3y1wVCtzJ1f2QDTuVzAsvazZqI4+pNB4LaAb+JPNQ+4
BtrQSXADXv7HfkUakzeZpgnJYw+zHWaVogKjcLDKHWdrbOECQQDpH/G+GsJ4mnrp
wZF9OxKqKhqBO73ZONHDxu55AukLghGnFh1udqdCQ7EPsaCqLN82RS4gn/WDfnBh
WB8DRavxAkEA3o6nMOMyKdsuqBbGyEPvaPDVmw973wtEohIj6MgwdYSUOhdKAurR
hs09yVGy0QpjoNHIE0vi5lUhPxJ1+Xvv1wJBAL0Ry14DFfX6U/WBqB2I63pW62gk
q7ShAH9nt8EtOxS6SNbaeMQ+Nyjm/ZNc3JEoE2BQezi6gsRCp6JLdduRhgECQD1p
V7EhwCHUnVc8kbWJKXLnocmbyC6PyWx/XPFK7DRBVTWCX6XWbeKol7gJlzIfj8Y8
nNzWP9IXA4mH6o3hKRkCQA+1er++Tx24uypEijIi7OK0bfjJUlrhCM9NVWxDKrzO
3zpuUB7yzuxrbcMZI8JKQIHL0sWz7egscepxS+N61y8=
-----END RSA PRIVATE KEY-----
```

Kumiko's certificate

```
-----BEGIN CERTIFICATE-----
MIICzjCCAjegAwIBAgIIAZUAcQIzAFcwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwMjAzMTg0OTIzWhcNMDgwMjAzMTg0OTIzWjBiMQswCQYD
VQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2Ux
DjAMBgNVBAoTBXNpcGl0MRswGQYDVQQDFBJrdW1pa29AZXhhbXBsZS5uZXQwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANX6dhOhUuf+2I3lymzeuSDwHLZqMqnu
3ISiIji/VlEoVUBFIHYjtxbmhIi40mEl4cqT+tVI6gY6Pe7VrL835Yr3AoLLeUB7
4mXa7T152+jAxA4+nCVnIAkMrPxTDeBFEfn+qyCRPWyQ7WEgH3Vd9AufnC7aeafD
pp+dcAOFZ2pBAgMBAAGjfzB9MFEGA1UdEQRKMEiGFnNpcDprdW1pa29AZXhhbXBs
ZS5uZXSGFWltOmt1bWlrb0BleGFtcGxlLm5ldIYXcHJlczprdW1pa29AZXhhbXBs
ZS5uZXQwCQYDVR0TBAIwADAdBgNVHQ4EFgQUNi5qQQ2G6AsiZK79cPEXYmPsqFIw
DQYJKoZIhvcNAQEFBQADgYEABIFd9N/3/05AD7Kt9kKSdy6vFvncU1IaccuFfdXc
QPfewY8NWwYKWsu588D4Nu77VQ++6a8AtjlJPSY/742Z4oKq1jfxdA+Uz/Z9cv2v
6aM4oX7R5FTgJTbHRC0ueH32OhNlcLhSNGHzNWSrS8AbtNOlfLRJipZI3N0W5b6q
09Q=
-----END CERTIFICATE-----
```

Kumiko's private key

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDV+nYToVLn/tiN5cps3rkg8By2ajKp7tyEoiI4v1ZRKFVARSB2
I7cW5oSIuNJhJeHKk/rVSOoGOj3u1ay/N+WK9wKCy3lAe+Jl2u09edvowMQOPpwl
ZyAJDKz8Uw3gRRH5/qsgkT1skO1hIB91XfQLn5wu2nmnw6afnXADhWdqQQIDAQAB
AoGBANJktWrxyanxC47iLdpEWHVJgoHeA7jQ8yS6orl3cPDVnpVWIufmkCTFPfWM
/Namv89HF3BVhD3hUHogwP03gcsIdxpccnu1wnmTW7IhSQXjBts0mEDbOw8S+WtS
9NjRI4m1+86OflE+TVa3DtwCE/pEOKhFvcZHvXiosYMnucABAkEA6xqKEwR1zI/V
u2B28Lcv0iafkJQDfPB3ooahQ+9qy5qUWgGZzXj6tM8YUusVqR/NCg8auqRC5uWD
yonN98phQQJBAOj/Pp9yyO2NCVs4Mp5QSXDOlRAOuruMz6vlmURQO/8uBmHvETfC
nkvqxxHjHW7mmusEY+ZIvRxmFV4RZcYByQECQHiT5/TQ+Mmti2TKmLXkffY+MOAp
yZAulG0at2LsS82YvjVbVNJ5Fbvd6w+72iQfVz2teXv3+wgI9orOGoDXnwECQGrE
I58PCzGHkkUBkHhpE+4kS7wK89hjYvpDAKOEHKoHHhecZAhoHv9suwHgT6l09IJD
BcANjtLHmHz9feRpBwECQQCuIn02CMxFy5yhjj4nlmCRQ6w6KBWjY68xnN4Qj/g3
SV+1HtmCclS0bK7e/IV6gOKn+MV3C+14JGdSRM+9HqcZ
-----END RSA PRIVATE KEY-----
```

Certificate for example.com

```
-----BEGIN CERTIFICATE-----
MIICjDCCAfWgAwIBAgIIAZUAcQIzAFUwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwMjAzMTg0OTA4WhcNMDgwMjAzMTg0OTA4WjBbMQswCQYD
VQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2Ux
DjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLmNvbTCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEA5jF2tSfMjTKFVnD3wjMzMiZCXjxocXsfeVDQcic7
Sq/yztEMvMBfMWpD53ytZL3H5iWfqs0tkKpohGJ7Bb5Dpa+76p2pW6RTnSKL2pYu
Hz+SRrjMyCQ8Rs1dLWSFsaTKAfGOxX4P/wCRo+rLPhICdaS7CMjQKu+zu3J6mOX/
n4ECAwEAAaNEMEIwFgYDVR0RBA8wDYILZXhhbXBsZS5jb20wCQYDVR0TBAIwADAd
BgNVHQ4EFgQUIurLOGYd8ZYMmke2uxxSRLB3ZY0wDQYJKoZIhvcNAQEFBQADgYEA
rutJ7R7xjSapbQOCktXfRMQeHwd1iDfkdpc1ElmYeXgWbjuxwCvbhQJrdMlbGZLa
fvVBC7zS3UWqb74k3EhXZtkugt+ejXADc3Xvj3pWTMxCvTFFsF7/0TvEgu79p8EQ
NOuBSRprhn7HYR2zuQoCvYT4R6/P8ahzqDEdIHoGf6w=
-----END CERTIFICATE-----
```

Private key for example.com

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDmMXa1J8yNMoVWcPfCMzMyJkJePGhxex95UNByJztKr/LO0Qy8
wF8xakPnfK1kvcfmJZ+qzS2QqmiEYnsFvkOlr7vqnalbpFOdIovali4fP5JGuMzI
JDxGzV0tZIWxpMoB8Y7Ffg//AJGj6ss+EgJ1pLsIyNAq77O7cnqY5f+fgQIDAQAB
AoGBANtRm2FkRv7seJ/wSA6OS6PnUeqJMZWVklo6xi9M86/oTbYA9VrNCqWBMqtW
XboTG2dKx4KrtFMWGTiwv7esHLPsUB1jYF7/KEsRh4WoRxfeWoQlAY6VYXycg6b5
X0uORdFMWL+WRxPmo8IhDKEwNyRyCyGQjfKpMj0724WjEqWxAkEA9MFDUQD+fL3N
ImRQl9ns3nHIIbcrtfxGCFaj+EJEwsyc5gq7QxRc3niNVt5pogPP7+CxskLaPPKU
TJmhtwixLQJBAPDE7hcDCPtsn9DIOXf/ZxXjfZAlAfwVsT+ggWQi5r63lGwjIbCT
qO6TijtbSqqD0QqULTabVwpIdYyknQqQlCUCQGnkG322UmQhsdiJUh0Amex7ibyc
hPrNVHdTFMnZ0en9oHwedHpHGw7dVTkaLNV9lL8RlY+sQMNRqDuj1EVeK1kCQQCH
945FLI+b/OHbs9bQb0k10TyNdHjEdTOdrPSlKhiIx39n+gcCgsC5ylQb5RgrZzlb
8gX+eocS5YyMmkGdP7yJAkEAsmGKAgt4nTfZY5L8PytPK8lCJjBLcyIllI3QEiMY
K/81YWYQcqsg5/cLBZC26KgNvxkyLwxS220Djlm19HJKGQ==
-----END RSA PRIVATE KEY-----
```

Certificate for example.net

```
-----BEGIN CERTIFICATE-----
MIICjDCCAfWgAwIBAgIIAZUAcQIzAFYwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwMjAzMTg0OTExWhcNMDgwMjAzMTg0OTExWjBbMQswCQYD
VQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2Ux
DjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLm5ldDCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEA2w4I/bz/vxzVskUEF56EYjf4yUftpG8jhmIiwsA8
AKLwc7CTnceW+tLmdDfUQLWw+HP4ky0tgQQA6pmviPORUNjuSj91dE7EJk3ZKePE
3MZ2M5JL6CEFn3HEFnHOQKv3TMKIGSpUZJjHmm15yRPiAlx0Q2vJ29h4W52X1DPM
62MCAwEAAaNEMEIwFgYDVR0RBA8wDYILZXhhbXBsZS5uZXQwCQYDVR0TBAIwADAd
BgNVHQ4EFgQUHNoIc7Or6o1iTsM1PmWPdgbxUAwwDQYJKoZIhvcNAQEFBQADgYEA
VlSod7+XfvSKNsybqtWPaM8VnoRLFVXvukgQbsdv4wuv5bnDfwxdU25rdizBbql+
m8Us+ky8ORw190v73mSeOro7KMv0mN1u2BaGUB/wjaRsH2HC+UZb0ok3vzZ+W8Re
ECjcVyHNRGVw5Iu2W5iWcO/a/74vPaVBiFQQJBRSLxg=
-----END CERTIFICATE-----<
```

Private key for example.net

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDbDgj9vP+/HNWyRQQXnoRiN/jJR+2kbyOGYiLCwDwAovBzsJOd
x5b60uZ0N9RAtbD4c/iTLS2BBADqma+I85FQ2O5KP3V0TsQmTdkp48TcxnYzkkvo
IQWfccQWcc5Aq/dMwogZKlRkmMeabXnJE+ICXHRDa8nb2HhbnZfUM8zrYwIDAQAB
AoGBAIrUP1CIutEldi3wXaKWfTI+ZPc0FeFz6mDdy0gAS0bf/WJk03lYqFA434Ni
aqvEOu+LmEu2gzNUFTyZwE0ciMg3NQ0H57z7OvbnHa0LajiJROo7zkROrmE5GTIV
v2WstOKJYsMdcTVa4VZd9cHH6zWXHtWDT+Y2MxrIerFnOYxBAkEA72cBQSE4SStZ
KvodDuMjFXG97Z1F927Xe/47iWnYRKhVB/jwN9uYpJog2cQFgsIsRMltozi3huTP
L8IKkI5N4QJBAOo95ShiRPcbXIXY1IcUGx1Rulr+paIAJwjuuutwrtCA1CbIKB0j
vfGVr3mKBGV2XLmz15nNV+5WFiLRBiUgucMCQQCxf+63KnlADurS6ZTH5/KoQKfw
WE568WzFWy8raBXYefJpsdHxqFiZmklHDIaFd5A5BBvNDA1O77EKGNWablghAkEA
zbvpPqv4+LRuchy8pZtyKTE0JWHNZlkN79mGEO4ajITqUNmx6c4PsVUQFwayz87C
qFQdxDdHyMyRiqjd5dQ1cwJAfJsXNGcOhilkV3xBy95tb3IsVP6G5DqwtID4hrYa
Onf9xrVzh9M29Xp+AHcwS4Y0+UgiNrd5BlbZs+ALZPD/jw==
-----END 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.

(dump removed pending verification)

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

Disclaimer of Validity

Copyright Statement

Acknowledgment

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