Example Call Flows Using Session Initiation Protocol (SIP)
Security Mechanisms

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.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6216.
Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Certificates .................................................. 4
  2.1. CA Certificates ........................................... 4
  2.2. Host Certificates ......................................... 8
  2.3. User Certificates ......................................... 10
3. Call Flow with Message Over TLS .............................. 12
  3.1. TLS with Server Authentication ............................ 12
  3.2. MESSAGE Transaction Over TLS ............................. 13
4. Call Flow with S/MIME-Secured Message ......................... 15
  4.1. MESSAGE Request with Signed Body ......................... 15
  4.2. MESSAGE Request with Encrypted Body ...................... 20
  4.3. MESSAGE Request with Encrypted and Signed Body .......... 22
5. Observed Interoperability Issues ............................. 27
6. Additional Test Scenarios ....................................... 29
7. Acknowledgments ............................................... 31
8. Security Considerations ........................................ 32
9. References .................................................... 32
  9.1. Normative References ..................................... 32
  9.2. Informative References ................................... 34
Appendix A. Making Test Certificates .............................. 35
  A.1. makeCA script ............................................ 36
  A.2. makeCert script ........................................... 40
Appendix B. Certificates for Testing ............................... 42
  B.1. Certificates Using EKU .................................... 42
  B.2. Certificates NOT Using EKU ............................... 51
  B.3. Certificate Chaining with a Non-Root CA .................. 58
Appendix C. Message Dumps ......................................... 64
1. Introduction

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

SIP with TLS ([RFC5246]) implementations are becoming very common. Several implementations of the S/MIME ([RFC5751]) portion of SIP ([RFC3261]) 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 hindered due to the lack of a commonly trusted certification authority to sign the certificates used in the events. 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 and private key that can be used to set up a mock Certification Authority (CA) that can be used during the SIP interoperability events. Certificate requests from the users will be signed by the private key of the mock CA. The document also provides some hints and clarifications for implementers.

A simple SIP call flow using SIPS URIs and TLS is shown in Section 3. The certificates for the hosts used are shown in Section 2.2, and the CA certificates used to sign these are shown in Section 2.1.

The text from Section 4.1 through Section 4.3 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 2.3. These host certificates are signed with the same mock CA private key.

Section 5 presents a partial list of items that implementers should consider in order to implement systems that will interoperate.

Scripts and instructions to make certificates that can be used for interoperability testing are 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 document that can be used for testing appear in Appendix C.
2. Certificates

2.1. CA Certificates

The certificate used by the CA to sign the other certificates is shown below. This is an X.509v3 ([X.509]) certificate. Note that the X.509v3 Basic Constraints in the certificate allows it to be used as a CA, certification authority. This certificate is not used directly in the TLS call flow; it is used only to verify user and host certificates.

Version: 3 (0x2)
Serial Number:
   96:a3:84:17:4e:ef:8a:4c
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
   OU=Sipit Test Certificate Authority
Validity
   Not Before: Jan 27 18:36:05 2011 GMT
   Not After : Jan  3 18:36:05 2111 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: (2048 bit)
   Modulus (2048 bit):
      5b:c2:de:0b:26:65:d0:91:c7:70:4b:70:0a:4a:bf:
      2c:78:ec:a5:0f:be:9c:10:ff:c0:0b:0d:73:99:9e:
      29:c3
Exponent: 65537 (0x10001)
X509v3 extensions:
   X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:

X509v3 Basic Constraints:
CA:TRUE
Signature Algorithm: sha1WithRSAEncryption

The certificate content shown above and throughout this document was rendered by the OpenSSL "x509" tool. These dumps are included only as informative examples. Output may vary among future revisions of the tool. At the time of this document’s publication, there were some irregularities in the presentation of Distinguished Names (DNs). In particular, note that in the "Issuer" and "Subject" fields, it appears the intent is to present DN's in Lightweight Directory Access Protocol (LDAP) format. If this was intended, the spaces should have been omitted after the delimiting commas, and the elements should have been presented in order of most-specific to least-specific. Please refer to Appendix A of [RFC4514]. Using the "Issuer" DN from above as an example and following guidelines in [RFC4514], it should have instead appeared as:

Issuer: OU=Sipit Test Certificate Authority,O=sipit,L=San Jose, ST=California,C=US

The ASN.1 ([X.683]) parse of the CA certificate is shown below.

0:1= 949 cons: SEQUENCE
4:1= 669 cons: SEQUENCE
8:1= 3 cons: cont [ 0 ]
10:1= 1 prim: INTEGER :02
13:1= 9 prim: INTEGER :96A384174EEF8A4C
24:1= 13 cons: SEQUENCE
26:l= 9 prim: OBJECT :sha1WithRSAEncryption
37:l= 0 prim: NULL
39:l= 112 cons: SEQUENCE
41:l= 11 cons: SET
43:l= 9 cons: SEQUENCE
45:l= 3 prim: OBJECT :countryName
50:l= 2 prim: PRINTABLESTRING :US
54:l= 19 cons: SET
56:l= 17 cons: SEQUENCE
58:l= 3 prim: OBJECT :stateOrProvinceName
63:l= 10 prim: UTF8STRING
43 61 6c 69 66 6f 72 6e-69 61                     California
75:l= 17 cons: SET
77:l= 15 cons: SEQUENCE
79:l= 3 prim: OBJECT :localityName
84:l= 8 prim: UTF8STRING
53 61 6e 20 4a 6f 73 65-                          San Jose
94:l= 14 cons: SET
96:l= 12 cons: SEQUENCE
98:l= 3 prim: OBJECT :organizationName
103:l= 5 prim: UTF8STRING
73 69 70 69 74 sipit
110:l= 41 cons: SET
112:l= 39 cons: SEQUENCE
114:l= 3 prim: OBJECT :organizationalUnitName
119:l= 32 prim: UTF8STRING
53 69 70 69 74 20 54 65-73 74 20 43 65 72 74 69   Sipit Test Certi
66 69 63 61 74 65 20 41-75 74 68 6f 72 69 74 79   ficate Authority
153:l= 32 cons: SEQUENCE
155:l= 13 prim: UTCTIME :110127183605Z
170:l= 15 prim: GENERALIZEDTIME :21110103183605Z
187:l= 112 cons: SEQUENCE
189:l= 11 cons: SET
191:l= 9 cons: SEQUENCE
193:l= 3 prim: OBJECT :countryName
198:l= 2 prim: PRINTABLESTRING :US
202:l= 19 cons: SET
204:l= 17 cons: SEQUENCE
206:l= 3 prim: OBJECT :stateOrProvinceName
211:l= 10 prim: UTF8STRING
43 61 6c 69 66 6f 72 6e-69 61                     California
223:l= 17 cons: SET
225:l= 15 cons: SEQUENCE
227:l= 3 prim: OBJECT :localityName
232:l= 8 prim: UTF8STRING
53 61 6e 20 4a 6f 73 65-                          San Jose
242:l= 14 cons: SET
244:l= 12 cons: SEQUENCE
246: l=   3 prim:      OBJECT            :organizationName
73 69 70 69 74 sipit
251: l=   5 prim:      UTF8STRING
73 69 70 69 74 sipit
258: l=  41 cons:    SET
260: l=  39 cons:     SEQUENCE
262: l=   3 prim:      OBJECT            :organizationalUnitName
73 69 70 69 74 20 54 65-73 74 20 43 65 72 74 69 Sipit Test Certi
66 69 63 61 74 65 20 41-75 74 68 6f 72 69 74 79 ficate Authority
301: l=  290 cons:   SEQUENCE
305: l=  13 cons:    SEQUENCE
307: l=   9 prim:     OBJECT            :rsaEncryption
318: l=   0 prim:     NULL
320: l=  271 prim:    BIT STRING
00 30 81 01 0a 02 81 01-00 00 ab 1f 91 61 f1 1c .0.........a..
c5 cd a6 7b 16 9b b7 14-79 e4 30 9e 98 d0 ec 07 ...{...y.0.....
b7 bd 77 d7 d1 f5 5b 2c-e2 ee e6 b1 b0 f0 85 fa ...w...[]
a5 bc cb cc cf 69 2c 4f-fc 50 ef 9d 31 2b c0 59 ......,O,P,1+.Y
ea fb 6f 1f 55 a7 3d-fd 70 d2 56 db 14 99 17 ..do.U.=.p.V....
r2 70 ac 26 f8 34 41 70-d9 c0 03 91 6a ba d1 11 .p.&.4Ap....j.
8f ac 12 31 de b9 19 70-8d 5d a7 7d 8b 19 cc 40 ...1...p.}...@
3f ae ff de 1f db 94 b3-46 77 6c ae ae ff 3e d6 ?........FwI...
84 5b c2 de 0b 26 65 d0-91 c7 70 4b c7 0a 4a bf .[...&e...pK..J.
c7 97 04 dd ba 58 47 cb-e0 2b 23 76 87 65 c5 55 ......XG.+#v.e.U
34 10 ab 27 1f 1c f8 30-3d b0 9b ca a2 81 72 4c 4.'0=......rL
bd 60 fe f7 21 fe 0b db-0b db e9 5b 01 36 d4 28 .'..........[6.
15 6b 79 eb d0 91 1b 21-59 b8 0e aa bf d5 b1 6c .ky...!!!
70 37 a3 3f a5 7d 0e 95-46 f6 f6 58 67 83 75 42 p7.?].F.Xg.uB
37 18 0b a4 41 39 b2 2f-6c 80 2c 78 ec a5 0f be 7...A9./l..x....
9c 10 f8 c0 0b 0d 73 99-9e 0d d7 87 9f cb cc 45 ...s...P.E
34 23 49 41 85 22 24 ad-29 c3 02 03 01 00 01 4#IA."$.)

595: l=  80 cons:    cont [ 3 ]
597: l=  78 cons:    SEQUENCE
599: l=  29 cons:    SEQUENCE
601: l=   3 prim:     OBJECT            :X509v3 Subject Key Identifier
606: l=  22 prim:     OCTET STRING
04 14 95 45 7e 5f 2b ea-65 98 12 91 04 f3 63 c7 ..E=-+.e......c.
68 89 58 16 77 27 h.X.w'
630: l=  31 cons:    SEQUENCE
632: l=   3 prim:     OBJECT            :X509v3 Authority Key Identifier
637: l=  24 prim:     OCTET STRING
30 16 80 14 95 45 7e 5f-2b ea 65 98 12 91 04 f3 0....E=-+.e.....
63 c7 68 9a 58 16 77 27- c.h.X.w'
663: l=  12 cons:    SEQUENCE
665: l=   3 prim:     OBJECT            :X509v3 Basic Constraints
670: l=   5 prim:     OCTET STRING
30 03 01 01 ff 0....
677: l=  13 cons:    SEQUENCE
2.2. 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.

Version: 3 (0x2)
Serial Number:
  96:a3:84:17:4e:ef:8a:4f
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
        OU=Sipit Test Certificate Authority
Validity
  Not Before: Feb  7 19:32:17 2011 GMT
  Not After : Jan 14 19:32:17 2111 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: (2048 bit)
  Modulus (2048 bit):
    00:dd:74:06:02:10:c2:e7:04:1f:bc:8c:b6:24:e7:
The example host certificate above, as well as all the others presented in this document, are signed directly by a root CA. These certificate chains have a length equal to two: the root CA and the host certificate. Non-root CAs exist and may also sign certificates. The certificate chains presented by hosts with certificates signed by
non-root CAs will have a length greater than two. For more details on how certificate chains are validated, see Sections 6.1 and 6.2 of [RFC5280].

2.3. User Certificates

User certificates are used by many applications to establish user identity. 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 a Common Profile for Instant Messaging (CPIM) gateway. In this example, example.com is the domain for fluffy. The message could be coming from any host in *.example.com, and the address-of-record (AOR) in the user certificate would still be the same. The others are shown in Appendix B.1. These certificates make use of the Extended Key Usage (Edu) extension discussed in [RFC5924]. Note that the X509v3 Extended Key Usage attribute refers to the SIP OID introduced in [RFC5924], which is 1.3.6.1.5.5.7.3.20.

Version: 3 (0x2)
Serial Number: 96:a3:84:17:4e:ef:8a:4d
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit, OU=Sipit Test Certificate Authority
Validity
Not Before: Feb 7 19:32:17 2011 GMT
Not After : Jan 14 19:32:17 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=fluffy
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (2048 bit)
Modulus (2048 bit):
00:a3:2c:59:0c:e9:bc:e4:ec:d3:9e:fb:99:02:ec:
05:61:0b:0a:ca:ca:ec:51:ec:53:6e:3d:2b:00:80:
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:

X509v3 Authority Key Identifier:

X509v3 Key Usage:
Digital Signature, Non Repudiation, Key Encipherment

X509v3 Extended Key Usage:
E-mail Protection, 1.3.6.1.5.5.7.3.20

Signature Algorithm: sha1WithRSAEncryption

Versions of these certificates that do not make use of EKU are also included in Appendix B.2
3. Call Flow with Message Over TLS

3.1. TLS with Server Authentication

The flow below shows the edited SSLDump output of the host example.com forming a TLS [RFC5246] 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 [RFC5246]. 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 SSL and TLS [EKR-TLS]. For more details on the SSLDump tool, see the SSLDump Manual [ssldump-manpage].

This example does not use the Server Extended Hello (see [RFC5246]).

New TCP connection #1: example.com(50738) <-> example.net(5061)
1 1 0.0004 (0.0004)  C>SV3.1(101)  Handshake
ClientHello
  Version 3.1
  random[32]=
    4c 09 5b a7 66 77 eb 43 52 30 dd 98 4d 09 23 d3
    ff 81 74 ab 04 69 bb 79 8c dc 59 cd c2 1f b7 ec
  cipher suites
   TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
   TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
   TLS_DHE_RSA_WITH_AES_256_SHA
   TLS_RSA_WITH_AES_256_CBC_SHA
   TLS_DSS_RSA_WITH_AES_256_SHA
   TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
   TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
   TLS_DHE_RSA_WITH_AES_128_CBC_SHA
   TLS_RSA_WITH_AES_128_CBC_SHA
   TLS_DHE_DSS_WITH_AES_128_CBC_SHA
   TLS_ECDHE_RSA_WITH_DES_192_CBC3_SHA
   TLS_ECDH_RSA_WITH_DES_192_CBC3_SHA
   TLS_RSA_WITH_DES_192_CBC_SHA
   TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
   TLS_DHE_RSA_WITH_DES_CBC_SHA
   TLS_DHE_DSS_WITH_DES_CBC_SHA
   TLS_RSA_EXPORT_WITH_DES_CBC_SHA
   TLS_RSA_WITH_RC4_128_SHA
   TLS_ECDHE_RSA_WITH_RC4_128_SHA
   TLS_RSA_WITH_RC4_128_SHA
   TLS_RSA_WITH_RC4_128_MD5
   TLS_DHE_RSA_WITH_DES_CBC_SHA
   TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
   TLS_RSA_WITH_DES_CBC_SHA
   TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
   TLS_DHE_DSS_WITH_DES_CBC_SHA
   TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
   TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

3.2. MESSAGE Transaction Over TLS

Once the TLS session is set up, the following MESSAGE request (as defined in [RFC3428] 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 [RFC4475] is used to break long lines. The actual message does not contain the line breaks contained within those tags.
When a User Agent (UA) goes to send a message to example.com, the UA can see if it already has a TLS connection to example.com and if it does, it may send the message over this connection. A UA should have some scheme for reusing connections as opening a new TLS connection for every message results in awful performance. Implementers are encouraged to read [RFC5923] and [RFC3263].

The response is sent from example.net to example.com over the same TLS connection. It is shown below.

SIP/2.0 200 OK
Via: SIP/2.0/TLS 192.0.2.2:15001;
branch=z9hG4bK-d8754z-c785a077a9a8451b-1---d8754z-;
rport=50738

To: <sips:kumiko@example.net:5061>;tag=0d075510
From: <sips:fluffy@example.com:15001>;tag=1a93430b
Call-ID: OTZmMDE2OWNlYTVjNDkzYzBhMWRlMDU4NDExZmU4ZTQ.
CSeq: 4308 MESSAGE
Content-Length: 0
4. Call Flow with S/MIME-Secured Message

4.1. MESSAGE Request with Signed Body

Below is an example of a signed message. The values on the Content-Type line (multipart/signed) and on the Content-Disposition line have been broken across lines to fit on the page, but they are not broken across lines in actual implementations.

MESSAGE sip:kumiko@example.net SIP/2.0
Via: SIP/2.0/TCP 192.0.2.2:15001;
    branch=z9hG4bK-d8754z-3a922b6dc0f0ff37-1---d8754z--;
    rport=50739
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=ef6bad5e
Call-ID: N2NiZjI0NjRjNDQ0MTY1NDRjNWNmMGU1MDA2MDRhYmI.
CSeq: 8473 MESSAGE
Accept: multipart/signed, text/plain, application/pkcs7-mime,
application/sdp, multipart/alternative

Content-Type: multipart/signed;boundary=3b515e121b43a911;
micalg=sha1;protocol="application/pkcs7-signature"
Content-Length: 774

Hello!

Content-Type: application/pkcs7-signature;name=smime.p7s
Content-Disposition: attachment;handling=required;
filename=smime.p7s

***************
* BINARY BLOB 1 *
***************

--3b515e121b43a911--
It is important to note that the signature ("BINARY BLOB 1") is computed over the MIME headers and body, but excludes the multipart boundary lines. The value on the Message-body line ends with CRLF. The CRLF is included in the boundary and is not part of the signature computation. To be clear, the signature is computed over data starting with the "C" in the "Content-Type" and ending with the "!" in the "Hello!".

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

Hello!

Following is the ASN.1 parsing of encrypted contents referred to above as "BINARY BLOB 1". Note that at address 30, the hash for the signature is specified as SHA-1. Also note that the sender’s certificate is not attached as it is optional in [RFC5652].

```
0  472: SEQUENCE {
 4   9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15  457:   [0] {
19  453:     SEQUENCE {
23   1:       INTEGER 1
26  11:       SET {
28   9:         SEQUENCE {
30   5:           OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
37   0:           NULL
39   11:         SEQUENCE {
41   9:           OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
43   0:           }
52  420:         SEQUENCE {
56  416:           SEQUENCE {
60   1:             INTEGER 1
63  125:             SEQUENCE {
65  112:               SEQUENCE {
67   3:                 SEQUENCE {
71   2:                   OBJECT IDENTIFIER countryName (2 5 4 6)
76   2:                     PrintableString ’US’
79   0:                     }
80   0:                     }
82   3:               SEQUENCE {
84   2:                 OBJECT IDENTIFIER
86   3:                   stateOrProvinceName (2 5 4 8)
89   0:                   UTF8String ’California’
```

:     
:     
101 17:   SET { 
103 15:     SEQUENCE { 
105  3:       OBJECT IDENTIFIER localityName (2 5 4 7) 
110  8:       UTF8String 'San Jose' 
:       
:     } 
120 14:   SET { 
122 12:     SEQUENCE { 
124  3:       OBJECT IDENTIFIER 
127  3:         organizationName (2 5 4 10) 
129  5:       UTF8String 'sipit' 
:       
:     } 
136 41:   SET { 
138 39:     SEQUENCE { 
140  3:       OBJECT IDENTIFIER 
143  3:         organizationalUnitName (2 5 4 11) 
145 32:       UTF8String 'Sipit Test Certificate Authority' 
:       
:     } 
179  9:   INTEGER 00 96 A3 84 17 4E EF 8A 4D 
:   
190  9:   SEQUENCE { 
192  5:     OBJECT IDENTIFIER sha1 (1 3 14 3 2 26) 
199  0:     NULL 
:   } 
201 13:   SEQUENCE { 
203  9:     OBJECT IDENTIFIER 
206  9:       rsaEncryption (1 2 840 113549 1 1 1) 
214  0:     NULL 
:   } 
216 256: OCTET STRING 
219  9:     74 4D 21 39 D6 E2 E2 2C 30 5A AA BC 4E 60 8D 69 
222  9:     A7 E5 79 50 1A B1 7D 4A D3 C1 03 9F 19 7D A2 76 
225  9:     97 B3 CE 30 CD 62 4B 96 20 35 DB C1 64 D9 33 92 
228  9:     96 CD 28 03 98 6E 2C 0C F6 8D 93 40 F2 88 DA 29 
231  9:     AD 0B C2 0E F9 D3 6A 95 2C 79 6E C2 3D 62 E6 54 
234  9:     A9 1B AC 66 DB 16 B7 44 6C 03 1B 71 9C EE C9 EC 
237  9:     4D 93 B1 CF F5 17 79 C5 C8 BA 2F A7 6C 4B DC CF 
239  9:     62 A3 F3 1A 1B 24 E4 40 66 3C 4F 87 86 BF 09 6A 
242  9:     7A 43 60 2B FC D8 3D 2B 57 17 CB 81 03 2A 56 69 
245  9:     81 82 FA 78 DE D2 3A 2F FA A3 C5 EA 8B E8 0C 36 
248  9:     1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50 
251  9:     94 75 7C BD D9 0B DD AA FA 36 E3 EC E4 A5 35 46
SHA-1 parameters may be omitted entirely, instead of being set to NULL, as mentioned in [RFC3370]. The above dump of Blob 1 has SHA-1 parameters set to NULL. Below are the same contents signed with the same key, but omitting the NULL according to [RFC3370]. This is the preferred encoding. This is covered in greater detail in Section 5.

```plaintext
0 468: SEQUENCE {
4 9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15 453: [0] {
19 449:   SEQUENCE {
23 1:     INTEGER 1
26 9:     SET {
28 7:       SEQUENCE {
30 5:         OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)

37 11:     SEQUENCE {
39 9:       OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
50 418:     SET {
54 414:       SEQUENCE {
58 1:         INTEGER 1
61 125:       SEQUENCE {
63 112:         SEQUENCE {
65 11:           SET {
67 9:             SEQUENCE {
69 3:               OBJECT IDENTIFIER countryName (2 5 4 6)
74 2:                 PrintableString ‘US’

78 19:           SET {
80 17:             SEQUENCE {
82 3:               OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
87 10:                 UTF8String ‘California’

99 17:           SET {
```
SEQUENCE {
  OCTET STRING LOCAL_NAME (2 5 4 7) "San Jose"
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER ORG_NAME (2 5 4 10) "sipit"
  }
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER ORG_UNIT_NAME (2 5 4 11) "Sipit Test Certificate Authority"
  }
}

INTEGER 00 96 A3 84 17 4E EF 8A 4D

SEQUENCE {
  OBJECT IDENTIFIER SHA1 (1 3 14 3 2 26)
}

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

OCTET STRING
  74 4D 21 39 D6 E2 E2 2C 30 5A AA BC 4E 60 8D 69
  A7 E5 79 50 1A B1 7D 4A D3 C1 03 9F 19 7D A2 76
  97 B3 CE 30 CD 62 4B 96 20 35 DB C1 64 D9 33 92
  96 CD 28 03 98 6E 2C 0C F6 8D 93 40 F2 88 DA 29
  AD 0B C2 0E F9 D3 6A 95 2C 79 6E C2 3D 62 E6 54
  A9 1B AC 66 DB 16 B7 44 6C 03 1B 71 9C EE C9 EC
  4D 93 B1 CF F5 17 79 C5 8A 2F A7 6C 4B DC CF
  62 A3 F3 1A 1B 24 E4 40 66 3C 4F 87 86 BF 09 6A
  7A 43 60 2B FC D8 3D 2B 57 17 CB 81 03 2A 56 69
  81 82 FA 78 DE D2 3A 2F FA A3 C5 EA 8B EE 0C 36
  1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50
  94 75 7C BD D9 0B DD AA FA 36 E3 EC E4 A5 35 46
  BE A2 97 1D AD BA 44 54 3A ED 94 DA 76 4A 51 BA
  A4 7D 7A 62 BF 2A 2F F2 5C 5A FE CA E6 B9 DC 5D
  EA 26 F2 35 17 19 20 CE 97 96 4F 72 9C 72 FD 1F
  68 C1 6A 5C 86 42 F2 ED F2 70 65 4C C7 44 C5 7C
4.2. MESSAGE Request with Encrypted Body

Below is an example of an encrypted text/plain message that says "Hello!". The binary encrypted contents have been replaced with the block "BINARY BLOB 2".

MESSAGE sip:kumiko@example.net SIP/2.0
Via: SIP/2.0/TCP 192.0.2.2:15001;
    branch=z9hG4bK-d8754z-c276232b541dd527-1---d8754z-;
    rport=50741
Max-Forwards: 70
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=7a2e3025
Call-ID: MDYyMDhhODA3NWE2ZjEyYzAwOTZ1MjExNWI2ZWQwZGM.
CSeq: 3260 MESSAGE
Accept: multipart/signed, text/plain, application/pkcs7-mime,
    application/sdp, multipart/alternative
Content-Disposition: attachment;handling=required;
    filename=smime.p7
Content-Transfer-Encoding: binary
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
    name=smime.p7m
Content-Length: 565

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

Following is the ASN.1 parsing of "BINARY BLOB 2". Note that at address 454, the encryption is set to aes128-CBC.
SEQUENCE {
  INTEGER 0
  SET {
    SEQUENCE {
      INTEGER 0
      SET {
        SEQUENCE {
          SET {
            OBJECT IDENTIFIER countryName (2 5 4 6)
            PrintableString 'US'
          }
        }
      }
    }
  }
  SET {
    OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
    UTF8String 'California'
  }
  SET {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    UTF8String 'San Jose'
  }
  SET {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    UTF8String 'sipit'
  }
  SET {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    UTF8String 'Sipit Test Certificate Authority'
  }
  INTEGER 00 96 A3 84 17 4E EF 8A 4E
}
In the example below, some of the header values have been split across multiple lines. Where the lines have been broken, the `<allOneLine>` convention has been used. This was only done to make it fit in the RFC format. Specifically, the application/pkcs7-mime Content-Type line is one line with no whitespace between the "mime:" and the "smime-type". The values are split across lines for formatting, but are not split in the real message. The binary
encrypted content has been replaced with "BINARY BLOB 3", and the binary signed content has been replaced with "BINARY BLOB 4".

MESSAGE sip:kumiko@example.net SIP/2.0
Via: SIP/2.0/TCP 192.0.2.2:15001;
    branch=z9hG4bK-d8754z-97a26e59b7262b34-1---d8754z-;
    rport=50742
</allOneLine>
Max-Forwards: 70
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=379f5b27
Call-ID: MjYwMzdjYTY3YWRkYzgzMjU0MG14Mzc2Njk1YzJ1NzE.
CSeq: 5449 MESSAGE
</allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
    application/sdp, multipart/alternative
</allOneLine>
Content-Type: multipart/signed;boundary=e8df6e1ce5d1e864;
    micalg=sha1;protocol="application/pkcs7-signature"
</allOneLine>
Content-Length: 1455
--e8df6e1ce5d1e864
Content-Type: application/pkcs7-mime;smith-type=enveloped-data;
    name=smime.p7m
</allOneLine>
Content-Disposition: attachment;handling=required;
    filename=smime.p7
</allOneLine>
Content-Transfer-Encoding: binary

*************************
* BINARY BLOB 3 *
*************************
--e8df6e1ce5d1e864
Content-Type: application/pkcs7-signature;name=smime.p7s
</allOneLine>
Content-Disposition: attachment;handling=required;
    filename=smime.p7s
</allOneLine>
Content-Transfer-Encoding: binary

*************************
* BINARY BLOB 4 *
*************************
Below is the ASN.1 parsing of "BINARY BLOB 3".

0  561: SEQUENCE {
4    9:   OBJECT IDENTIFIER envelopedData (1 2 840 113549 1 7 3)
15  546:   [0] {
19  542:    SEQUENCE {
23  1:       INTEGER 0
26  409:       SET {
30  405:         SEQUENCE {
34  1:           INTEGER 0
37  125:           SEQUENCE {
39  112:             SEQUENCE {
41   11:               SET {
43    9:                 SEQUENCE {
45    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
50    2:                   PrintableString 'US'
55  19:                 }
75  17:               }
77  15:               SET {
79    3:                 OBJECT IDENTIFIER
84    :                   stateOrProvinceName (2 5 4 8)
89  10:                 UTF8String 'California'
94  14:               }
96  12:               SET {
98    3:                 OBJECT IDENTIFIER
103    :                   organizationName (2 5 4 10)
108    5:                   UTF8String 'sipit'
114    41:                 }
110  41:                 SET {
112  39:                 SEQUENCE {
114     3:                   OBJECT IDENTIFIER
119     :                   organizationalUnitName (2 5 4 11)
124  32:                   UTF8String 'Sipit Test Certificate Authority'
127  20:                 }
130  20:               }
132  20:             }
134  112:           }
136  102:         }
138  94:       }
142  86:     }
144  75:    }
146  63:   }
148  54:   }
150  45:   }
152  38:   }
154  30:   }
156  23:   }
158  19:   }
160  15:   }
162  12:   }
164  9:    }
166  4:    }
168  0:   }
170  5:   }
172  1:   }
174  0:  }
176  0: }
153 9: INTEGER 00 96 A3 84 17 4E EF 8A 4E
164 13: SEQUENCE {
166 9: OBJECT IDENTIFIER
: rsaEncryption (1 2 840 113549 1 1 1)
177 0: NULL
179 256: OCTET STRING
: 49 11 0B 11 52 A9 09 9D E3 AA FB 86 CB EB 12 CC 8E
: 96 9D 85 3E 80 D2 7C C4 9B B7 81 4B B5 FA 13 80
: 6A 6A B2 34 72 D8 C0 82 60 DA B3 43 F8 51 8C 32
: 8B DD D0 76 6D 9C 46 73 C1 44 A0 10 FF 16 A4 83
: 74 85 21 74 7D E0 FD 42 C0 97 00 82 A2 80 81 22
: 9C A2 82 0A 85 F0 68 EF 9A D7 6D 1D 24 2B A9 5E
: B3 9A A0 3E A7 D9 1D 1C D7 42 CB 6F A5 81 66 23
: 2B 00 7C 99 6A B6 03 3F 7E F6 48 EA 91 49 35 F1
: FD 40 54 5D AC F7 84 EA 3F 27 43 FD DE E2 10 DD
: 63 C4 35 4A 13 63 OB 6D 0D 9A D5 AB 72 39 69 8C
: 65 4C 44 C4 A3 31 60 79 B9 A8 A3 A1 03 FD 41 25
: 12 E5 F3 F8 47 CE 8C 42 D9 26 77 A5 57 AF 1A 95
: BF 05 A5 E9 47 F2 D1 AE DC 13 7E 1B 83 5C 8C C4
: 1F 31 BC 59 E6 FD 6E 9A B0 91 EC 71 A6 7F 28 3E
: 23 1B 40 E2 C0 60 CF 5E 5B 86 08 06 82 B4 B7 DB
: 00 DD AC 3A 39 27 E2 7C 96 AD 8A E9 C3 B8 06 5E
: }
439 124: SEQUENCE {
441 9: OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
452 29: SEQUENCE {
454 9: OBJECT IDENTIFIER
: aes128-CBC (2 16 840 1 101 3 4 1 2)
465 16: OCTET STRING
: 88 9B 13 75 A7 66 14 C3 CF CD C6 FF D2 91 5D A0
: }
483 80: [0]
: 80 0B A3 B7 57 89 B4 F4 70 AE 1D 14 A9 35 DD F9
: 1D 66 29 46 52 40 13 E1 3B 4A 23 E5 EC AB F9 35
: A6 B6 A4 BE C0 02 31 06 19 C4 39 22 7D 10 4C 0D
: F4 96 04 78 11 85 4E 7E E3 C3 BC B2 DF 55 17 79
: 5F F2 4E E5 25 42 37 45 39 5D F6 DA 57 9A 4E 0B
: }
: }
: }

Below is the ASN.1 parsing of "BINARY BLOB 4".

0  472: SEQUENCE {
  4  9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15 457:   [0] {
19 453:     SEQUENCE {
23  1:       INTEGER 1
26  11:     SET {
28  9:       SEQUENCE {
30  5:         OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
37  0:         NULL
  :   
  :   
39  11:     SEQUENCE {
41  9:       OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
  :   
52 420:     SET {
56 416:       SEQUENCE {
60  1:       INTEGER 1
63 125:       SEQUENCE {
65 112:         SEQUENCE {
67  9:           SET {
71  3:           OBJECT IDENTIFIER countryName (2 5 4 6)
76  2:             PrintableString 'US'
  :   
  :   
80  19:           SET {
82 17:             SEQUENCE {
84  3:             OBJECT IDENTIFIER
  :               stateOrProvinceName (2 5 4 8)
89 10:             UTF8String 'California'
  :   
  :   
101 17:           SET {
103 15:             SEQUENCE {
105  3:             OBJECT IDENTIFIER localityName (2 5 4 7)
110  8:             UTF8String 'San Jose'
  :   
  :   
120 14:           SET {
122 12:             SEQUENCE {
124  3:             OBJECT IDENTIFIER
  :               organizationName (2 5 4 10)
129  5:             UTF8String 'sipit'
  :   
  :   
136 41:           SET {
5. Observed Interoperability Issues

This section describes some common interoperability problems. These were observed by the authors at SIPit interoperability events. Implementers should be careful to verify that their systems do not introduce these common problems, 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. See Section 26.2.1 of [RFC3261].

Many SIP clients were found to accept expired certificates with no warning or error. See Section 4.1.2.5 of [RFC5280].

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 checks that this name corresponds to the identity the server is trying to contact. Normative text describing path validation can be found in Section 7 of [RFC5922] and Section 6 of [RFC5280]. 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 will typically 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 will need to match according to the certificate validation rules in Section 23 of [RFC3261] and Section 6 of [RFC5280].

Some implementations used binary MIME encodings while others used base64. It is advisable that implementations send only binary and are prepared to receive either. See Section 3.2 of [RFC5621].

In several places in this document, the messages contain the encoding for the SHA-1 digest algorithm identifier. The preferred form for encoding as set out in Section 2 of [RFC3370] is the form in which the optional AlgorithmIdentifier parameter field is omitted. However, [RFC3370] 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 [RFC4134]. 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 SHA-1, 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 [RFC3853].
Observed S/MIME interoperability has been better when UAs did not attach the senders’ certificates. Attaching the certificates significantly increases the size of the messages, which should be considered when sending over UDP. Furthermore, the receiver cannot rely on the sender to always send the certificate, so it does not turn out to be useful in most situations.

Please note that the certificate path validation algorithm described in Section 6 of [RFC5280] is a complex algorithm for which all of the details matter. There are numerous ways in which failing to precisely implement the algorithm as specified in Section 6 of [RFC5280] can create a security flaw, a simple example of which is the failure to check the expiration date that is already mentioned above. It is important for developers to ensure that this validation is performed and that the results are verified by their applications or any libraries that they use.

6. Additional Test Scenarios

This section provides a non-exhaustive 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 both related normative works such as [RFC4474] (particularly, Section 13.4 Domain Names and Subordination) and informative works such as [RFC2818], Section 3.1. To summarize, test plans should:

- For S/MIME secured bodies, ensure that the peer’s URI (address-of-record, as per [RFC3261], Section 23.3) appears in the subjectAltName of the peer’s certificate as a uniformResourceIdentifier field.

- For TLS, ensure that the peer’s hostname appears as described in [RFC5922]. Also:
  * ensure 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. See Section 7.1 of [RFC5922].
  * ensure that the most specific CommonName in the Subject field matches 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 ‘*’. The peer’s hostname is discovered from the initial DNS query in the server location process [RFC3263].

- IP addresses can appear in subjectAltName ([RFC5280]) of the peer’s certificate, e.g., "IP:192.168.0.1". Note that if IP addresses are used in subjectAltName, there are important ramifications regarding the use of Record-Route headers that also need to be considered. See Section 7.5 of [RFC5922]. Use of IP addresses instead of domain names is inadvisable.

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 subjectAltName)
3. S/MIME : Bad peer certificate (valid authority chain does not end at a trusted CA)
4. S/MIME : Bad peer certificate (incomplete authority chain)
5. S/MIME : Bad peer certificate (the current time does not fall within the period of validity)
6. S/MIME : Bad peer certificate (certificate, or certificate in authority chain, has been revoked)
7. S/MIME : Bad peer certificate ("Digital Signature" is not specified as an X509v3 Key Usage)
8. TLS : Good peer certificate (hostname appears in dNSName in subjectAltName)
9. TLS : Good peer certificate (no dNSNames in subjectAltName, hostname appears in Common Name (CN) of Subject)
10. TLS : Good peer certificate (CN of Subject empty, and
    subjectAltName extension contains an IPAddress stored in the
    octet string in network byte order form as specified in
    RFC 791 [RFC0791])

11. TLS : Bad peer certificate (no match in dNSNames or in the
    Subject CN)

12. TLS : Bad peer certificate (valid authority chain does not end
    at a trusted CA)

13. TLS : Bad peer certificate (incomplete authority chain)

14. TLS : Bad peer certificate (the current time does not fall
    within the period of validity)

15. TLS : Bad peer certificate (certificate, or certificate in
    authority chain, has been revoked)

16. TLS : Bad peer certificate ("TLS Web Server Authentication" is
    not specified as an X509v3 Key Usage)

17. TLS : Bad peer certificate (Neither "SIP Domain" nor "Any
    Extended Key Usage" specified as an X509v3 Extended Key Usage,
    and X509v3 Extended Key Usage is present)

7. 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 SIP for Instant Messaging and Presence
Leveraging Extensions (SIMPLE) Instant Messaging and Presence
Protocol (IMPP) agent from www.sipimp.org. The TLS flow dumps were
and TLS" [EKR-TLS] 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.
8. 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 any messages to check certificate revocation status (see Sections 3.3 and 6.3 of [RFC5280]) as that is not part of the SIP call flow. The expectation is that revocation status is checked regularly to protect against the possibility of certificate compromise or repudiation. For more information on how certificate revocation status can be checked, see [RFC2560] (Online Certificate Status Protocol) and [RFC5055] (Server-Based Certificate Validation Protocol).

9. References

9.1. Normative References


9.2. 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 CAs 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. OpenSSL version 0.9.8j was used to generate the certificates used in this document. 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 certificates 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, e.g., "makeCert host.example.net domain eku". 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 Privacy Enhanced Mail (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 Public-Key Cryptography Standards (PKCS) #12 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 (version 8), Outlook Express (version 6), and Firefox (version 3.5) can import and export .p12 files and .p7c files. You can convert a PKCS #7 certificate to PEM format with "openssl pkcs7 -in cert.p7c -inform DER -outform PEM -out cert.pem".

The private key can be converted to PKCS #8 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, a single file with both certificate and private key PEM sections, or a single .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

```bash
#!/bin/sh
set -x
rm -rf demoCA
mkdir demoCA
mkdir demoCA/certs
mkdir demoCA/crl
mkdir demoCA/newcerts
mkdir demoCA/private
# This is done to generate the exact serial number used for the RFC
echo "4902110184015C" > 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 ${CONF} ] && break
done
```
CONF=${OPENSSLDIR}/openssl.cnf

if [ ! -f $CONF ]; then
    echo "Can not find file $CONF - set your OPENSSLDIR variable"
    exit
fi

cp $CONF openssl.cnf

cat >> openssl.cnf <<EOF
[ sipdomain_cert ]
subjectAltName=${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment
extendedKeyUsage=serverAuth,1.3.6.1.5.5.7.3.20

[ sipdomain_req ]
basicConstraints = CA:FALSE
subjectAltName=${ENV::ALTNAME}
subjectKeyIdentifier=hash

[ sipuser_cert ]
subjectAltName=${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment
extendedKeyUsage=emailProtection,1.3.6.1.5.5.7.3.20

[ sipuser_req ]
basicConstraints = CA:FALSE
subjectAltName=${ENV::ALTNAME}
subjectKeyIdentifier=hash

[ sipdomain_noeku_cert ]
subjectAltName=${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment

[ sipdomain_noeku_req ]
basicConstraints = CA:FALSE
subjectAltName=${ENV::ALTNAME}
subjectKeyIdentifier=hash
EOF
cat > demoCA/private/cakey.pem <<EOF
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFDjBABgkqhkiG9w0BBQowMzAbBgkgqhkiG9w0BBQowGgYDVR0P
BjowcywN0x0Q4wDQYJKoZIhvcNAQEBBQAD saludable
basicConstraints=CA:FALSE
subjectAltName=$\{ENV::ALTNAME\}
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment
-----END ENCRYPTED PRIVATE KEY-----
EOF

Jennings, et al.              Informational                    [Page 38]
cat > demoCA/cacert.pem <<EOF
------ BEGIN CERTIFICATE ------
MIIDtTCCAp2gAwIBAgIJAjajhBdO74pMMA0GCSqGSIb3DQEBAQUHMAwGCCsGAQUFBwEB
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA5pCBfGd0j5IAv5oMTG92MKXgKksq3WksX7jY4pFRxf9wJt4DQ
-----END RSA PRIVATE KEY-----
EOF

# uncomment the following lines to generate your own key pair

# openssl req -newkey rsa:2048 -passin pass:password \\
  -passout pass:password -set_serial 0x96a384174eef8a4d \\
  -shal -x509 -keyout demoCA/private/cakey.pem \\
  -out demoCA/cacert.pem -days 36500 -config $(CONF) <<EOF

# US
# California
# San Jose
# sipit
# Sipit Test Certificate Authority
#
# EOF

# either randomly generate a serial number, or set it manually

echo 96a384174eef8a4d > demoCA/serial

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

# Make a symbolic link to this file called "makeUserCert"
# if you wish to use it to make certs for users.

# ExecName=$(basename $0)
#
# if [ ${ExecName} == "makeUserCert" ]; then
#   ExtPrefix="sipuser"
# elif [ ${ExecName} == "makeEkuUserCert" ]; then
#   ExtPrefix="sipuser_eku"
# elif [ ${ExecName} == "makeEkuCert" ]; then
#   ExtPrefix="sipdomain_eku"
# else
#   ExtPrefix="sipdomain"
# fi

if [ $# == 3 ]; then
  DAYS=36500
elif [ $# == 4 ]; then
  DAYS=$4
else
  echo "Usage: makeCert test.example.org user|domain eku|noeku [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

ExtPrefix="sip"${2}

if [ $3 == "noeku" ]; then
  ExtPrefix=${ExtPrefix}_noeku
fi

DOMAIN='echo $1 | perl -ne '(print "$1\n" if (/\w+\..*/))'  
USER='echo $1 | perl -ne '(print "$1\n" if (/\w+\..*/))'  
ADDR=$1

Jennings, et al. Informational [Page 40]
if [ $2 == "user" ]; then
    CNVALUE=$USER
else
    CNVALUE=$DOMAIN
fi

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

case ${ADDR} in
  *:*;) ALTNAME="URI:${ADDR}" ;;  
  *) ALTNAME="DNS:${DOMAIN},URI:sip:${ADDR}" ;; 
esac
echo rm -f demoCA/index.txt
touch demoCA/index.txt
echo rm -f demoCA/newcerts/*

export ALTNAME

echo openssl genrsa -out ${ADDR}_key.pem 2048
openssl req -new -config openssl.cnf -reqexts ${ExtPrefix}_req \  
  -sha1 -key ${ADDR}_key.pem \  
  -out ${ADDR}.csr -days ${DAYS} <<EOF
    US
    California
    San Jose
    sipit
    ${CNVALUE}

EOF

if [ $DAYS == 0 ]; then
    openssl ca -extensions ${ExtPrefix}_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 ${ExtPrefix}_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 \n    -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; \n            mv ${ADDR}_cert.pem user_cert_${ADDR}.pem ;;
      *)    mv ${ADDR}_key.pem domain_key_${ADDR}.pem; \n            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.

B.1.  Certificates Using EKU

These certificates make use of the EKU specification described in
[RFC5924].
-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pNMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTA1VTMRMwEQQQQQDAPD4YXwp2m9ybmlhMREwDwYDVQQHDAY4g3m9zTGE0
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpccGl0IFRlc3Qg2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAxDTExMDIwNzE5MzIzXNloYDZixMTEwMTE0MTkzMe3WjBQMqsw
CQQYDVQQGEwJVUzETMBEGA1UECwMUEweAwIBDQYJKoZIhvcNAQEBBQQwWgYDVQQK
U1NTMzE2M1Y3MDMwQ0EYDVQQLweedM1Y3MDMwQ0EYDVQQLweedM1Y3MDMwQ0EYDVQQLwde
-----END CERTIFICATE-----

Jennings, et al.            Informational                    [Page 43]
Fluffy’s private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKBQAYQAEoyxZD0m850ZtvnuZAuYxNjg3q0x1Nzwq2r1c9X1UID4x+pOnw
aDEojMyVnHdIpenbf2nIoBta9hvV9+wRT1mCm0NE44liubDdrbJ7141qysBHgy
mMJ152q3LbM84+u+P+LWUJQF3v+p4GvN6fn4x+wjdbLyL2wUQsYBaOGP/V2Nfc8W
3KoFYQsKxsrsUexTbjz0rAID+NRsGCMtIaET88z9Kw60ogug14QUu4r4y/BGp6G
mhqokxX+kBjtNWDwb2/wj/jaxwB3MstMYxsiy0Y+iyoArv634bR1BIx1H+GUv
i/slnhNERjTXJBzPVo3UvtaUW/Cm2+Pdz7Ty1QIDAQABoIIBAH+bSvjiQ1r1WnnW
YM78s4mpWeDr5hrvjmMQsyu/zQe11u4551T9Fgc0l1DQGtpFJLaTz5U4g4yjVq
3QG61eL5mKfddDH2R+z13SwuMmYQG2ZTaZ41IVdo+V/v8Ap+T9yH4A2Ug1wQSoA/3
R0PLN3ITaw8neH+wiaGgsweujBvcaIJu4RQuGHRHaeEplU+tffcHHE1fzUAmKym
CMQFG8ipdUcAlpHy3Pyc0oGnLyEvnv291xGWQfWT7nqf7K0QQLA6+TvbG3fGEYIw
WK4DMraUb2661JnjiXfAoxWOTysygV+KyhZcbwjbWAUSOSduAtfwa6b720nWd28J
8KyrvXECgYEaCJZSSavxhlxqxsWC/dWdQ8S3SimI62KSlrN3i10/R0/60KUI2ap3
162hNlQ8t3DjpkWizruki2sOdUs7k3z6g+qm++POTUwL7z3Bri0FimqUeVSyGaf
ZmFgG7wLAM29zvho0T2jGrwrM1NSyJ2t+jyqpiO1XqkbdbPBPxKrdCqGEA/Y0W9f
4M2KQKQBFzjecPeQpwJqnh8cuOS+2CNYLJgmjzd/AUgVF2+WPA1R1DmJaqJ9iwh
15Yx3CbkphKbphf1ilMhkCyGaf+jqAisq/NzN3YaOPF9W9tho0FosAhT95XfWxH6
YBKUrQpOP5D4AY427EL1nsIRA+LtoPaTqpphFzMCgeYElqSOOh2s2FA43uyTpeF3t
rmQvQaIaB7KFSaigGBqUY7p0k0F9dWvRT419sd48a7kb09ur2K08heZ8BenoB
Oj+Hiy靳NjHSRjNqNBLuTP2fMU+PDFX/92n6WFjXkXB+d1PV8SJXkUJgiC36/H
1uHmZxQZFBBXXPQTRQG3GdcCGyEAoPFqs8QZOIA+BbnzqvI8OzfuN8geFy9JurSm
55pXxT0hB2xst3tDjMBZG15KUu9nbW6gb/PVBbStSTv6vdtDKpyq7OaAgACyC
Zv5PAFhF0v79NACsHIX2C1drU7EJaFQ3nA4pFhef7NaK5haG78yFPoqUIvP
0i0XhxtCycGEA0xU1k+5XwItRnc1FU0t6+4R0zCt7qSOEpQRktz/saNXEhA6N
EUqWLnMOoClhpb7V51v5xKxgjX8Glpg1ZehIIt5jz88XmmB1SQxVLTP6r3p8Fq1M
EtXf7h7dZjKURP7dO2uG4b0JMF5590nNjrxj9VeSxEWURSK3YGH/h8=
-----END RSA PRIVATE KEY-----
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pOMA0GCSqSIb3DQEBBQUAMHAAxCzAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxp2m9ybmhlhMREwDwYDVQQHDAhtTYW4gSmp9
ZTEOMAwGA1UEECgwFc21waXQxKTAnBgNVBAgMclFPW10MQ8wDQYDVQQDLDE4XzA
 MATERIAL 8GA1UEBxMIU2FiuIEpv
C2UXdjAmbNgNVBAoTBXNpcGl0MQ8wDQYDVQQDEwZrdW1pa28wggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwBgEAEKAcIuAQDl5odVj3GgFf/MuGIq5bMY8k17g7kUfexe
Kptxt1x2DHzU8X/PUN2WzXcTbp019DqA+MkMzX4GNgpD2yeoIrcquKUX7UQL0K
Y6a11DijH7qCqPTWFIrRhbRUHdJj0WvG1AFYYRGG/IZfRqCmH8Aw18XSp614m1n
9XwL5LuHNimAgjADHMMSk10mbWw0thU6nVotU1G1SA11A32JXZ81bqKq0d3Tq1ho
fsKU3GwoBZG5071VG5c2v2ByA5HnCFPFeDfTQZ23197USLhqrTGrxxr646F0e9
P0mYH6s31RveAzhDKibChGaqKg7r8+SZDnLdCy3DrFSPC/1baAgMBAAjGgocwcw
UQYDVR0BESzwSIYwc210mmt1bW1rb0BleGFtcGxlLm51dIYVaw06a3VtaWtvQG4
YW1wbGUMBumV0hhdwcmVz0m1bW1rb0BleGFtcGxlLm51dDAJBgNVHRMEAgABG0
1U0dDgQWwBQ02bNX/rnbbYoEY6wU7oyst36WbDAfBnNVHSMED2AwgBSVRX5fK+pl
mBKRPBNjx2iaWBZ3JzALBnG1NQ8EBAMCBeAwH1QYDVR01BBWyFA1KwYBBQHUAgwQ
CCcGQAQ6FbwMJUMAGCSqGSIb3DQEBAQUAIAA4BAQCT2SN7LjUvcgVnB13RBrRd0+p
aiFPWQ+YWWbJCG/+NetesegCw17xBFQGw+Gx1WpTVuDu55myTTyrMqpckcy0
KvuUcvz/yK670s5ume1vc757KqY8vqfeXZ5ZG4Pjqqe1J3czBOXlf65KfmlQLH/q
R4i/o9+hHB3zoe5b5m5f2g91jWyRbDiUuav/ah9WeGEXK992d9XJ/bgGPeAdgvm3jo
KDFKh8slyfim3VuD0uPots2nlzNGaooceeFZoYaMf8uZaoan6kXQQTDMRpt
YKxy5712re/840FWvDx767w+G1FF71SrAkHwroYt0NMnLv610rka8qYvaQ
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAy+aHVXQN4BX/zhiKmzGPCpe4O5FH3sVpKV206bcZcX9g/IC
1F/Pz1J9113E2z9NfQ6gPjJDII+1DTRqQ2cnqCK3KrilFyu1EUaCskOlWtdQ4oxO
6nT01h8K0YW0VIT449FxrTXQT2E1URvyGX0UHB/AMNcPF0qeteJtZmPV8C+S7hzYp
gIIwAxxzKpNaN5h8L1NLyvFZ1ldVBTUgNZQN9iWV/NW6i4gN06tR6H7ClNxsKAWR
ud05VRuW3FdgocGR5whaRXg0w2BNt9fe1i4abSkq8ca+uEhaPQ5z9JmB+nt5Ub
3gGYXSiGwh4BmiiQk+/kmQ5y3qsg6xUjw5WwIDAQABoIABHCMxrGgRS0xWLBW
PLBm0IhLRsri14+bqwb663SHTABI1Yzuw+W2Bo2oMnvjMrEe0o407J2J6boJ2zvF
CKmKqrYiKaJxgBr/JtZ6xCWGFCNAl1pnXLlW5g5DIgj8sAL0Q0N7hyR0rr4ArR
W0vuVQ5SYFFX4BhvdxEseyyRwCqn3x0PFSfF95Ad+vuJd5CYuFZCuyGkszQ3fi+Nia
Gqs01Euyo0Ev72rsW2E5+wtx3qX8B24HXR+yQ9NhBE81p2CDw1Uh1qIHL8kw8mNIG
V3oLKiIowV+M62X/uwwAMF0rdn5kE7+b5DO1IKsUAAa8LZsf95OvkLgw7aiz5e
sXHdAEGEcYEA8930YQU2+AczzjC5hygw1M/X5k/icc2pOa8/inn2jW7iZgQ0AFE
jjxuoIVbxBs9fC2+M6g76Svw99eBmovLArqbhbFaLfbZCsrLcEahGtGcu3w7o6px
N0EfBBF5FmOK7qaQ1SGqj0NF5zP2JsxrCNoMrFmFwVdcp/P/3Jp/I1zEsqYe1gXi
/7I8h9og1dmTPzmpvnpnANdRF/iuMXAE4LNRP09hxj0B7Vuat1ABtx0/9ZN1hLh
BTZ5R292RjzbSHXZ3FdcoGx9Q3qa+xuPe14RcppHNJykDphPLnOUwBqFL6kyU
nTEF+k6VIZvXmsGB6wphHU1cjDAZUX71p6W49TECgYAMHpa7pExUDT07rhrH9tpC
sume5441shtX0WobOAipVCulgxeDRkmBWIBW7YoUSyqH82JoPM81amfqWQJmZ9Yh
/5Y1AIw1J+k+9QVnZJlMmN6ohDvDFVqEm9VECE1HS/Mmox6FwZ8EJS7hAvAzzzy
Dqhtbh6wFW5WYm15zD3xewKBgQCRRm1ky/QGF0m0+H5ZMgB3e17GGLB1sNe0n1YvE
Dzv0pc3UHQGQ7TCLDuLY919C80t1+V76JIIHDYy97U4bdBau/kgGm+ngd9PJ
U1iXq8aam73rU3LXhW72H86R1A6jQtI4tpcNW5S/pr51n0UY1/hXTKT7psPIAzA08
0V1kQKBQ9aGzCyc/6wumGJUerVcZzd/H6+E3ntZmzz+3c8+wV896RFzzU0JY4
bVnrYFz91kFxlNtGNECEUZVZdXHUAgqge05rbzPudA24wSrsNChU0yw8LKLXHDckt
pVLs0vHJK2W/W2I+P2exSPQPT30y8tT6Is9ZbNg/H4D160heHkuQ==
-----END RSA PRIVATE KEY-----

Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pPMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTA1VTMRmWQYDVQQIDApDYWxpZm9ybmdlMTIwNjA0MTQwMjEzODEwMDA3MDQw
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIENvtm1yMREwDQYJKoZIhvcNAQEFBQAD
ggEAM4Qv6N23v7Qp0v2Q2h4F277777777777777777777777777777777777777
Jennings, et al. Informational [Page 47]

-----END CERTIFICATE-----
Private key for domain certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3XQGhDC5wQfvyi2JOeb1KNIN4WebYMSFAajkix+oaMp4C5
v1LspspjR4St9nSFghZ+TjZACQsIKIqlOmp/NC9wcWN96UNquzqR63TKMLChsh
DSpzB2O0jXW/S0S5mdgffqCnpshk+5REbBI2qKEjcfSML5GG0vW78EFDvgEW6
XoSXdsEgJcGSHYK91ViZPr0aaJ1TGtFCN1htU0WVLa3RIIZK5UD6fHfYdWyxKs
M3gAULoHBo5APrjvWzheyj9B674ao0haeDD31h6bLjs3Hips/ar+z6x+VxucE
1o28oEdUCyxaVoaLmcox/xP2EaJEN3f2FpSw1DAQBAoIABAB9s23In4ID40Wm
u7W48acCFLlsSLMzqoMEkWcN6FO4zDTc23LagaJxje0UMuuKXFEYWAAPr6RBCIM
yHQLQMoOCdLNX4y+d+2uUElLQg+9aAUu09ebDxcMntkf6yNysUS/mk/KQMPbpFRT
1dn80wXsJc19I6yXArkB7/9UEcUUt6Zvdbz+aqXpH2Z4TJe50WQZKXHsYobM8Y8
c2Xwud1zdQtvOtrOeirexpxQPqf4CBQnBx0Gmbae9f27Kw2bBm5+blZFdqnx0h
6Q3r9J9EDYWkrVMAq9a67a959ST1lymoC0c6FmToCMG1gOEPcUedvUNPw123220k
ZdfsawECgYEA+AewMiTDhAE+9T1d2qi1LQV+y8bdTHQ9rS6F9q5580pqq79E9
asuDuqkU+Tiew50icrrkIyzQmC1fnfBjH5y6GupkUK8HdLLkA29FV3Ze+Y4ZbL
b4TEy/RxEECQERgtqNia080lITd0bNwxzVsi3nrhtOphFBPBUZU3sCqYEA5J2G
aGRKcyxASGAn2mqXCP/p1Mu+7Jb2OcGQO/3gsxI/191LwRtRhgFs/pTyCg2Wlpb
+mpnQgexKtowldbjorrUAdWw84Zg4u9d+uWOXXEpCVIEu4DZsRURdy3Opk1vJaUm
NLqB1dJ8zkUfRXti4Rzx1Xysf6nWaxDPDdI+GEcGyEAoyFrYY+dohSvS9uijY4e
FV5n5t8E7QF7L72soDlhY1jDOV2+VFL7erbDusJ751q9hj1q7i3d1s/M87P
2qjsMtGbojRts0TS1v6mx16CD5Fmm/yfF1BeaMZ9FPnNq41p388SyPrhTbC7v7k
3aO7 AptXtw6VzBPVucz8A/8ECgYEAw2ps2F13qdql3ns01Ho3gqVoaGUUUI0KL2I
wJYMI/AkZrR4KthmlPIEPt/t0psB2y2BOk6X0yY5a+10WZ0yGHHJneJRe7RgQ9
hQ0EHEqUzQzrkR9Hu1rSpG11oMcmCQF4TBy+Vl7+YnKQAU1dP3G71qgIA6fjXe9l
ofZxG6ECgYEAyAHvSeqQwmdotcpoN3fK1CmRHTzFZtF2rsuRBAb0ynW69Ut9ote+
+Bmd4fUBUB9rQOzUXdFRoK3+wlJKHEPjm/0FxtCQ19o9En4e69jOwXJNKSa
GjGURqZV3m2baNeM7s8C5Q9nskDudlVAB2bMk23O16cvPIb0E=
-----END RSA PRIVATE KEY-----
Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pQMA0GCSqGSIb3DQEBBQUHAMHAxCzAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxpz29ybmNhMREwDwYDVQQHDAHTYW4gSm9zZTEO
MAwGA1UECgwFc21lwaQxKTAnBgNVBAoMFVBQdQwGCSqGSIb3DQEBCwUGCCsGAQUFBw
MDIzMQswCQYDVQQEExIyQ0EIBMlMgYQYDVQQIEwNleGFtcGxlLm5ldCExMDAQCgYD
VR0MKw0GA1UEChMIRW5pZm9ybmlhMQ8wDQYJKoZIhvcNAQEFBQADggEPADCCAQoC
BQAwHjEKBgUEv76t0q4Z5RcaPA5QYic1QTyXrV6EsQg5Zv/1N5V+jxIi4s0Xnk/3sG
Hr4o6eV6xuc3vJxYmR0zAX6cCZ4E/F4Dq85IEpV64p9yyG4PfK/Ml9I3eFHvN9hL
5G+6fWqSxhB68aFjC7XHsXpE0WwX6h+7w3I/1sEe9bG7Q42s8P80i1yHtP1zfAA/
-----END CERTIFICATE-----
Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA7Cx2A9VI8zB4tawEMdRJclLPWTqm7ogxWDwHYx/WDRNZM9zm
NQBCVaUs22Y31BxLorPnFV5y9/essyiYK9lp2t6S1NOUDPmXbBCDI/6/H8d/UEHKP
ezYMT8pqmgdCMVBrAccxxz6QK1Jrj0IV0YPZ2cD1xgNXE2YoYrP/7kdjF7oHiJo1kY
2VIHwAM9QtwWDPryE7Tom+sYW3WF20C0Lx/klfX5hegmpcGWpk5XxaqJqanpWWD3gu
u8NmtC01d+b+jsS4VUirfga+HdyGHOSQxtuT4g+BW/uiGNY8OXpKBQQTZCccG0F03g
xc8rN6dddfmWhHnZ83K037a0xxujhA99NmUavHQIADQABoIBAb8FBR2B1fTf1056
yLE6aSjWriILhD76NFxr/AlIg79MBwubEjCNIo2N5+cKxVxv4x219NOU0+t2TiIL3
KGyfKecO6isncjxKgn0nzw/o3n01z97Txpxb9mL9t3GHOYRoUvK6xGpGILo60B1Cz
F+8pk0jegc7eVF0UoPULHm/FCmpY30N5cvChcAE/ncW49b2mH3qG+cmr5UCkKDUY
baJyLd8Q1f+uSmtrfYYzRT5c+4wmrBUjv3w9poMuEc0s1RaDnyeKJPSNR/6/LjK
tqnggNifc9j9qf6fWA23dDmmU/kSrtn1KOz5XmV9Jbo4Fu6F4vn/m/hJ5og4CP9
hZUIQIEGcYEA+nVPzpsoCF7se8e7vNhCvQj0nJAliSqrMrqXQITSRm+Q0oQ6UT
PVN4GEOs8TTJyYvKpaoaqQ36VLv/Wr0jUm+2+dv1T1I1FTWJas8RNmZHMv0LfwEe
Qu2fTI612d/L9GBMUCYa/sucX5Fgq+3LC+Qo9j9w8eHw2QzQw321cWYER4dsQGyEA8WYX
AqDkDjHrQu2h248gZsuogiqz05iuZxhK2VT4q1M92mu8m1tak+eov/3wjoq4uw
TAQbf/t8EfQ7LIGjKaQaU7m7gA/aNB6MGwdpBAPUZDL+DuKfbDzBZTOL/IuaW0P
40RC0Ulp5nTU9wzIKB7a6n5550KXIXiGUpfhCcCgYa61YdPmziU0fxJ79zrUGyV
82KwWbzqxyplvqGzEhstSaRs4s9ZQiyLVIECIM252i01iRSU/r0ncPvEUC9gq
+Sg7Zkbh1643p4WuUbGmHGCsNhsO+3JHM/jagGlwmzbwVM7+MwtNT7IVhJ6u3J
EuUkJ1biHsXv5z3AbWekHwKkgBy5HwflCExhA62o9NhImPy28YquC1Q4tjReyu
MNz6AIayahZiITxBG08f9aAeDrxxyVzY5kIMI1EnFrPwF480DcpMPNInk1L1VO
kwBQqOidrods3jyyaZCzC7fJvXrxKUSUfdjw+b2A9kzhjv3HCg2qbl/5Utraio
JMMFaOAABh+k+C4e8wRw+Fxbb/DgAk50K5vzZLHeWBIG9BE1626N/0fEQVXp
zqwyNo6zQ0amfM56an16F2N7MC1SGJxh27eBT1Lp1NXC1GWAQEtXmnTvnAZNzXC
5U0wv55sLxOnbnhJWNN8BwzjHUp0KU3pn99GcF+vk5jEg7Zftg=
-----END RSA PRIVATE KEY-----
B.2. Certificates NOT Using EKU

These certificates do not make use of the EKU specification described in [RFC5924]. Most existing certificates fall in this category.

Fluffy’s user certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pRMA0GCSqGSIb3DQEBBQUAMHAxCzA JBgNV
BAYTA1VTMRwEwYDVQQIDApDYWxpZm9ybmlhMRMwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECzo.terminate.\n[...]
QXV0aG9yaXR5MCA3MDk\n[...]
-----END CERTIFICATE-----
```
Fluffy’s private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAulcjiD+lADV8u+uih4mA8cjqWBBW6Cyb+xhLcFFkGUT4ULq
xYnTreUIJArKW5c2jKLcu5Z1HSwxVCvChnWy1lhjHAdoa3syfuq8CTyv2xeWA4olbVa
jXEvE2z/yzUcs7xJ36lM1p897fiquI4GpOF04wS3j9kTtE+ux1tKRGxVbA3mLVvhtX
mwwEem3+1XW2tiug55pMU+Ua+qGCRyAKWG7m5NvGrAUEhBgQYDhQKUX1IKW46CUf
6wQv3tW8SvjjXRZVz5cIyPF0tzKB1G2oE4ydfVLvEcBDyVwmv2N5mD+/WLp9jzF
91EHDyHPwuglwbN8gFctK6DtRuguaVx/SjDTJQIDAQABaOIBAbtIBAbI+8K5eJlvw
/MoxOwKrwrw8E1fppGTxfhJN31MbfIIFAS5hjD3GncDqwaAMI3yks6YEz+mu/rdHH
wp2FXCXO1FgSedbd8TcmILbO27v0fXZUkTTh4aj4YH0YrLg7yfrySjXE8WQ1KPMK
PVkLMOpk34+2jOOhqUDpR3xhcJC1Q81fc1hKe2JoixNDpDf6m3aztq8QUQLQD2I
mjww1IH1677G5o/6cMloC0MTeqv/3cuWlRmvPv4eyGDnUuFXKFpB4DQQML7TD8
FodHBymHIOzSSF+gYgBFob0YNgu2CqZrfED9cf0rRotrbXf6tM+akclxHhKfKaa
JP2osBUcYeA4MaetKas7aEhYMc4TKhxhV5hI61jixr/6h+yuTy0IOJOb9MyU3
5n6vLyghNhBwKbK080OWP0F4syyvKyp2e1jmduraH29DKatRLEKUK082RG4AmXmk
G62sWOFx6JF350nAKyv/7aN9jc4K1v6EFYQGYExbp40IthFbBae28CqYEA1DmX
ikJD+jW9y9pHk51YJj+r+a5qPPNvmjGKQje3Y6+sLxm0WhXoCBOYrWhBRA/
Sxh39P2i8RcENknxhp6Ac087Z2Gco15U6Kh+rwd/3+SsHqPrugaDlWlnGkcuv9V8rP
8uP2ChjoDbi5UY2UR97GKV98x8k7Sf6kDT32mQMCyB/KH3R8V7j01KcqTc1UWl
J1E3/gB4S+wQ8EYELlh0FvCP0sdLzd1tFwR70fUraaO1k/SHeSlfiJDIghN6nz
oDFMq+7vhr47uWurZPc95n4nk51hKr1RN1nV9e1JTudjLcW3spoFCy2JUX1IOE+n
k6zuzzCFdUWFCs2jiubqwKBgC76RHe1JjkDe2FniX8r7D88/y/W9xXVtdWqgqE4x
XQ/OIP8A6iJbKTA5qScp2kBAssdPj7c7vEa21a8FvQPXVZCrsAAPXha41335sV0
WYBill17T7XAZ2yvdwV8w/Gnds00z0Ll1TRGx6W+sAYr0r/1/Mb/tOknASHeYEqEy
Y/w3AoGASjocCFyj2aBVHHSQaImK/Xk3h0FR4myOGWtHxkXmeozOY2ydcMO1drlz
A/qRvVoHFrwyoaik2kAlPrGxyEhCcmOs1h9yIACxfW2RfCq39Dy9RTrRkwa
ArJmcEdRESOS1YnHHXGfE1QMG1wVjUXMWeYctLtqQWk1LDDTYYFQE=
-----END RSA PRIVATE KEY-----
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHxAczAJBgnV
BAYTA1VTM4MwEQQVQIDAPDYWxp2m9ybm1hMREwDwYDVQQHDAHfTW4gSm9zZTEO
MAwGA1UECgwFc21waXQxKTAnBgNVBAsMIFNpcGl0IjIc3Qg2VydGlmaWNhdGUg
QXV0aG9yMRU5MCAxMDIzMDIzMjMzMjM2MQswCQYDVQQGEwJVUzETMBEGA1UEB
ME1UEwJwMTE0NTE0Nzg0MGEYDQYJKoZIhvcNAQEFBQADggEBAMAQGMBAAGj
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxP0FTe58Qw7uai+l+m9HJhyBR/dyoTW0xMgorIq4uDnlAYqr/XYEYDwV336yCFUHSHJqQa3jw7rB3bAvRHxVr4PqUZ1RAA4MHHDtBeQFW6tzCtjY5muSNJcKSGtEV2ENCv/6F//o2XmRbpRbcv0fcnq9Cmln5S0Xeir3CERos/axYZUwflZ40rzSVdf16J23L07i2496mXYW+61gSOD6EJV1XrcrvkswFXujLFOzcC5qUKSM0HUrCcsWgLFvglMsbp4V1oj2fPPFDOL142nGbo/BryfMN144CyULbPK1kJCwpBl2ke4K0wzhkPdHuvnIO4VrqBq2UANEdIfMI1kM+QIDAQABaoIBAduLR+kwp3sVr1cX341FsomBLNeKpA+4+K/JC7xQ9bfACXhec2AnuWLZ6TUNRFgoK12dVookYe
gHD57n36dcf9KR7rph5ixo0Rl1JncoiRFNeFpRNZiCZBwniAXFLnHGTznVnpwT7xi
axMNzrU6epi00/quAPkoU5x6e0+j+j3ZauI4EfD1w2R6moBMUaTAuZEEyLuC9A6bfZ2AFDchPVlwsJNm0tAjc8Fss8xKIs9HUXGS22eufHxFkCGwChwuW60obGmasE7G57h4q9QyQ9hGhV9/yMmQ88GM7LYnOyzFBCpuwjQTHwsD6741dMf4kXYXVjcnTAKcCgYEA4bjN2ILIs3uWTjvTNhrMwn1q0ZBDgh1LuNs5o1xT0J7cckUKvs
nqyQ0YznK/9N8uUs12s3csXhypuuGrJwAVf648RSPDUUQ2XOoPL9NeuZt5V1fT
1VvWanKCBZ5sztISNVpt7P8dtGRh4C87Z/7M+gEUYQ1Og7z7fJHvFszqCgYEA32m
61N67aHkgMLa60Z9JfK/zsFIPjpw24tk+sQCQezawPvkT7qF2+U81VtOXXKJL
axxospsULC8586TEAPoYjtj9k1i6Zzj8mgRZLU559+gRdTpAPxMgIc7tU7U6cDIw
SPA61uxJp9XCA/gf6yLfas9Vbhc/80C7I4ygjLdscgYEAqAG7yuM/CSY3MrArRw8f
f4W9qkgHtwwn2qjg0btkjek8GOKxvce4Q9aJoiy6HPZM8hpo6kIUsczyXGcKf
s33Yucz+0rZ4uqX3bI4tNf150P0F0En28KhXSJrmxbZxob+LMmJNUF6yiuL6W+cuQxAlilyeOGJes63hpl0130cQEAUEcILGQpTCMyAYWgC93n5u6i+IrIx08gyl
ewlihrahKiLWTyTsxyhQwKb4i10IWEHHv7DFPDhcs3tcIezhN86m7K7tAfjHO
YZfsmFU991utPwGKnwQq1Q0KqOkeR7cOLdSxRML5045uKJnYmmkSp7tHjJfN9XKe
q4fK40wCgYoAvtE1Mq7s3htx4hL2uM3FwLp/1mGWZDMMAzDW+XswwZ66rbcWY
8c3hbohuJbpjAzbaq4OR2G+qtRm0dLca+tQFMrObEThFg1NCY+WoHRSNR1mbCS8w
dsszgHWF1nrxBLdDF1hZ0wSqb2tLbyBP1HJF+FT1P06UT8aDQ4Pw==
-----END RSA PRIVATE KEY-----
Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAjAjahBdO74pTMA0GCSqGSIb3DQEBBQUHAMAxZcAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRMwDQYJKoZIhvcNAQEBBQAD
AQ==

-----END CERTIFICATE-----
Private key for domain certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAoRW5jJmVqp+qSz273JZBGn1z8j5WxCAakkfe8B9qy0SUJ3GU
FXv5HYJpD8adc0cTSUALKIpFy88erk4vHVwCMAMxaZxcG/ArmXM37sxQLL8wh1
yvsTY136nLw3dgYNYm15SWMChgcM7h/FL8EXLzOjMARGP885+kTjQQAEX1qP7T7G
O2QST2QzJsBxJkJkncgc/fAM/GYzMKw7D9+lXt/L2PUaJQO7PTUX2p8P6nP0rWyf
GBOUro2GW50bglrBKIFzVbSBF+P3/+FHtmB4HSA7ossZdW2d2Xzsl0hckxkK2
C3g/k/yx11hJBDnd1u3W1FSA2r76U+ySKKcoS5Q1DAQABoIBAB19g1ZAOedzLxJY
Cja/ON4EBBdRhLUumvOnccJc/J3XtD2Nnt8T0gdJUojDhjjwZQzz7kYdzD4Nj6
Akeszb30sT2MTdFy/WiCT6cA1H1VrrKZ3cK6zyY217aPj1H8IUaUr1T73U7nT/DMp6
mGFbo+XQZ18eFc8zubb+BK7KsN4N6b/zhmw+PXElug2EDG1NFo4TMhxPD4wBIMU
8oLLe8A6GKixmx3gMuIiI6Raau2HPGkjkhhkAax/yzUl88BCMLOldJjyyH19RtSri
n0VFef0ogMo2pzdZ/94ynFDFMnBXTqBabT09eicyuLKILog/ERmj6jImGShYRWed
GzlZXCUCyEAOFDUek2uLhyITXw1zjDT1duyIt1Yzg/Mexaq2eA96zhJID6xA+55
PQIxEfhehTNf4e4cKjXQSD7aixy7jp/kFGowFRI4pwbLduhniYxa8Kv0OpfJM4
DTAue4QF2Id5243KH755UbTj7rCEidQnij44DA3gPnjqKX973dpVCxGEAxfUX
/zMXgTp7hxW+QHZD7xXEs4Fp1xjzL5BaHoJnM7WbmkWvUvcaMAEA/i9rgypGXRIN
jX6KBZ9UVgh/B0/AcyMA3DImTa0+Uie9kn7jT5pzvIUAdF+RyQ4utULW5crgrzv
PjG99tXMTuhuIBLSumVEwvcP6Skilr4pxle1z8CgYEAFA51sk2c1qMqznXJjm
JbIdsa+w6ycD9mluqaGxGo8usmqcZ70KrsphEm0QFvFisjPnU271LZI1/AKcdVz
kedDUF5FxzT4Jd13B713xFRXYBp+3ZyVfNbovWSc1VncjRg8aMIa5mES8m
UfhtFnRPOPWmn6qmyQvJnTkCc1B/3z1linkBKq9ooZEU3Iq6TXL5pLemOl0FQjCk
kJvNvTRcXTM5pngPSEailp6QCO+s0OYg1nvyV0SwLPWV/VbBfDH3JzWC66vcKeuc
Dz5JnFWg5mLiIzbly/WTaocCh0J1FWi5j7iggHc9Uu0Ov0bqjY5e6a6+Hv4sNUO
ho1chQKBqCGKLEH7wQX8fkw+yKnmxAR05Z3H1H5UQY/Wfsc/OVnWnO+yovwvueTTt
cbw1Vkr4EjP9uFeE5MIZkmsNUJGrCpx/3uq2JFMVOpJzJN9b1FM4u1cQf91eh
hiVIFVMxq+dVMXBGxKhnYK1Mnt9b3B6mDqerQJk1TkryaqJ2QPq==
-----END RSA PRIVATE KEY------
Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pUMA0GCSqGSIb3DQEBBQUAMHAXczAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxp2m9ybnlhMREwDwYDVQQHDAYW4gSm9zZTEO
MAwGA1UECgwFc21iwaQxOTAnBgNVBAcMIFNpcGl0IFRlc3QgQXV0aG9yaXR5
MAwGA1UdDwQEAwIF4DANBgcqGAEwIF4DANBgkqhkiG9w0BAQUFAAOCAQEAJ
-----END CERTIFICATE-----
RFC 6216                  SIP Secure Call Flows               April 2011

Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAqhbHyDUpucZfZqQ5etuh3altHR+mRhZxs+7+0hYd12X4AiUwq
o08pKYr1rAAVtana5/tL3k6ReGSEbkDbG6B+1Q1LVMoGOpwJQ2A8sxq+T03kVr6N68H
DAGNxtA0tHp/qRZreuvbL8XDNoKgrD7xCbghuYfYds2psaWAIo5edi3gsabLon17
riF9DbWp7TN71x8sypkFuXM9XMyv61ucfbnMTQL+jvV7H799DIOowfQ034DVxkWsj
Y12V5UC4foXa/HHy3BlTNQ1fJd7MWbDlJHOIglarVs1c7QRE17uEicsK/2nr82814
exKsW4a0pJAgLqTvgsIPsFvmeW2ctWAns7G3bwIDAQABoIBAHJjpv+BSY5V1T5L9
+UCrfJyKVLGLiqOF/CygafjTzTVa6/v+a/r8Rkgb8XyrJ9sXvZVB1tiUbdM42z91
8faVSKLAsjs3thkfSojTMzU77x+iDcG6LxSzekaAGqAIJ7sRL+iEzl/Fm1WlgEYhl
GIWIIghH01n300eCy72dwmAV+02ahzn8eBggkWxm0pfb1RC9pVh0FC0+jy11HasJL
oOBkh511bmZ4PUuUY0Y72jz665gm7i0nr25igeF842JkbbqAV8rAo1q2Y7qY7LYEw
6QyLvoodeb0rH28IEzahWAdmIFPGICUcF7M7myInoataGQdVEUuYnkUQQVo1/JSX
46CCmbEcGyeAE41cdv/IVz29dwJo/0Ma9j4zqeg7Pgn5DRXnNjmCsSxVHSMINw1U1
BcYozs77vWbuiXu1X02q9e9mGA2ss3+VaNxBoe66E6bFK16cQQHQ52nXv1l3qknK
5BSelFKcZKpFVNWrq0BCC6csDndTcHp95TIKsWkeseCz3v5UXMoscCqGWEAUNV
+SaC1QGTL8Z8fkyE2nHqRUFXnkC/tWQjopg5nE4ws3Lqj13NYCuQR/sD4elxQDE3
6C0m197Jcz27jggD7g9ri1xMznRXLMeG7bb7f7fwpE/SKVOH5ugAE7btF18x1IJk
yOck1lul1lQjToSa4uetHLRXKCDSePrlSw7wRdkgCgYEAkDBKBYAynykerDqBi57
1phFD9DG5x+VPTU0x6WugpabF3jEANHzQg0oTdRTDrYm8T8dp22WiS3Ab7WS
hfcCtVwccM+ID29OnKvQ76lM66C72j36sEXBuPeeA072ZKZDc1xsdM3JnNn
+M2KxGxGxL19tIehJ3lfouykCqY9BAdU1PwaVeTVX13orduyhUQ0xOoNma491Eu8
FpciPdZtImzkyZWvJpeLwFqWlq1mMjzJeWnRFpWqCrRI5szqKs9jzj/wBnei12BM
ctTXLRp6vFnhJg+xo4eQ5H4s5KfYb1hHs51cUdx+p04sWEyxM+Gnd2FCxYsAEF
UCJ4QkBAKSrm85yXqhd8RMAg9JZLUGPmpTNUnU99f3UFe67jZ2EZEatsn18vd
65x04h58cohJnXqel6k31c3MwOzPSzrVh1a32MEmoJCPgwBav8Lzrrl13YQln6yf
+bAminFTdmhigPORB36ODY4B1kwkxKzQn3XAtlrl7N7RV5wHr2ejkY
-----END RSA PRIVATE KEY-----

B.3. Certificate Chaining with a Non-Root CA

Following is a certificate for a non-root CA in example.net. The certificate was signed by the root CA shown in Section 2.1. As indicated in Sections 4.2.1.9 and 4.2.1.3 [RFC5280], "CA" is set in Basic Constraints, and "keyCertSign" is set in Key Usage. This identifies the certificate holder as a signing authority.

Version: 3 (0x2)
Serial Number: 96:a3:84:17:4e:ef:8a:52
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit, OU=Sid Test Certificate Authority
Validity
Not After: Jan 14 20:21:13 2011 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
OU=Test CA for example.net, CN=example.net

Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   RSA Public Key: (2048 bit)
   Modulus (2048 bit):
       d4:41:3f:7e:2a:e4:26:d5:a3:33:b0:5e:37:1d:e5:
       3e:07:31:dc:97:c5:d6:19:26:bc:7d:0b:ff:8e:de:5e:
       f9:0f:d9:9a:45:0f:2b:8d:dd:fa:15:56:d5:35:17:
       f1:5c:5f:c8:4f:36:54:8a:b6:7b:6f:ff:85:5f:88:d8:
       d8:df:a9:7b:40:45:4c:92:0f:aa:b2:2c:da:8a:64:
   Exponent: 65537 (0x10001)

X509v3 extensions:
   X509v3 Basic Constraints:
      CA: TRUE
   X509v3 Subject Key Identifier:
   X509v3 Authority Key Identifier:

X509v3 Key Usage:
   Certificate Sign
   Signature Algorithm: sha1WithRSAEncryption
   c2:3e:00:ce:5f:b4:c8:da:a1:b5:2f:c2:89:60:a4:3a:2b:be:
Robert’s certificate was signed by the non-root CA in example.net:

Version: 3 (0x2)
Serial Number: 96:a3:84:17:4e:ef:8a:53
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
        OU=Test CA for example.net,
        CN=example.net
Validity
  Not After : Jan 14 20:21:13 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=robert
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (2048 bit)
  Modulus (2048 bit):
    b4:0f:03:3e:a0:00:d6:c3:26:e7:57:8e:21:92:a3:
    5f:3b:41:36:59:70:be:7:4f:08:6b:4a:db:44:
    34:e5:77:be:89:f1:c9:76:4c:37:34:3a:bc:99:4c:
    f9:23
  Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Alternative Name:
  URI:sip:robert@example.net, URI:im:robert@example.net,
  URI:pres:robert@example.net
X509v3 Basic Constraints:
  CA:FALSE
X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:

X509v3 Key Usage:
- Digital Signature, Non Repudiation, Key Encipherment

Signature Algorithm:
- sha1WithRSAEncryption

Certificate for CA for example.net in PEM format:

-----BEGIN CERTIFICATE-----
MIIDzzCCAregAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBAQUAA4IBgAwggEiMA0GCSqGSIb3DQEBAQY
MIIDzZCCB6rKAwIBAgIARh2x5GBfii8c3aqo22Jb7o0r7z3AhMB8GA1UdDgQGC3qGSIb3DQEJEwEg
MIIDzzCCBAkgAwIBAgIARh2x5GBfii8c3aqo22Jb7o0r7z3AhMB8GA1UdDgQGC3qGSIb3DQEJEwEg
-----END CERTIFICATE-----
Private key for CA for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA1EZlUfiEHHLTR6UVFABS3Cp3kxFedRTSiFS9f1DdT9+KuQm
laMzF43HeWNNxwcaYCK7/0ieNF008PelvuHMI18BhSAXf0r12Q+BzHc18XWGSa8
fQv431759yArQ8oI36FVbVNRCogNL89aVLUOLeQ4U639D190MZqI0LVNpI
2NyGQtumRISVAhS5T9KdpVJjcbY/M/Ay8hj8W1Rd3WXXxFXI9D21uirz7/b/h+NjY
3617QVMkg+qsiyhGqTVMsIEKhig20VRZD8DFKsRK2GElrBpgmuJbtJKzkQrVeW
K4cV2cIRAI7CCPC9V91E9PY51cvjTDpP156oOQ1DAQABoIBAdp/7/pJH79vnc3
z7hGNE50aGBHuPrSh3yJG4a+O67XbzARN213XzaiEhGixoY7dua97xtu4dbJc
f2Jj1r4uAISa4svV7NDwO9VNW3o8NkWdLEnV288Eo2Tgqc8wxZ/BleL9nCJWcH4Y
Jw1rKKWmTdQpVBCWCpil9UzduXqT2fBrsl6+OZ+F3k bvUwYAVhhUUBS9sf4Xib5
GA2CDLPm433giOS3yr9KigpCvAhAiMiPTX6i65m9xGQCjxhP/dorOHrCnzRD
y0F0CaNRUjg9keVw+nUg1aVFOnU7rgcb1FXg07ea7G+mfp3Cfm744kvFEFz04k
8WLW6gECyEAY1k9KmHMeB1+xPJB4a2a5QvrF7nL8e7/0aTNCyM10l3uxPyDPj
TNEfgaRobptmwd2HVtEjLQ54fE+pE+qS8DOORH2VFqWi91zI48WnmW/6j5P+qiXY
tc2ZPF22msWS7UaQyOhUfIMhxoi1BbUH5QYrca59mmPtaxcZ+IECgYEAAJ07
6DamIgy0eJ02GKHU/Hy8RvQ2gauzCttnmLQzW2omx9hORe1a71Q5F6Y3HOCRcTD
RDDDjua9Y8BJ0WTKasbgxJqH1q4pUd6ycfWgISbcNFTosgIP+/0ZPE4EKD10
rb1du1zHPuZD2Q72KtSPMK+ikny21C29cm2mKuCGYEAsoGoX4f/J/HPDMzrKf4gT
CoBojxZ2+wBvPTVF/0LwBwTCG3VrGpZ5GwO41RwpFqHmwwU9cmE+N2TJQKXLQ+4
7Vp1vy6vOsAM9SsCswOw2ZtBF1w4v0QP3W37AAaTUCgGFTKnbTjQjX/QF0aHb2c
6Kxszs5MqVz7j7FVypc51ECqYATq1WpHQc99Qv4sJUnu4v+dbJ6f9q9Q6qNf
HEU9GC2BC5Nwx7D4+rXm7qWNCst35Nn9mKL0RBGqq2RxoFuj7y71oXnuEU
BmWyGqj357Hv3aF0Nw/EzqEjT07v0xFgiUtgb4+VoaZHyy1BG8s7pjrCw7d
qD7L/QKlbGcEdLkx51ld/EwqW00N5qg/51g/T0uz3MCd1zCjfa2BMqsvNRAlD
unMMANDE1PHOF57fSmCf3pS6N7+iW1s/k91wupwQ7sT2Y8dSRYdpFjF9R8u25yX2
mdRbU3JxIdAqPEEkpBo1XpLoeLGvoTHFSazgmCPIKXq0wL+0+w==
-----END RSA PRIVATE KEY-----
Robert’s certificate:

-----BEGIN CERTIFICATE-----
MIIEJjCCAw6gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMH0xCzAJAJBgNV
BAYTA1VTRMNwEQQIewDPYWxpbmr9yMhMREwDwYDVQHYHwTYW4gSm9zZ2TE0
MAwGA1UECFMf21waXQIXDAegBNVAsTF1Rc3QgQ0EgZm9yIGV4Y1wbGUumVO
MRQWEdYDVQQDEwtleG1ctGxLlm5ldDAgFw0xMTAyMCcwMS0xMDQgMTE4MDEx
MIIwJExM1owVjELMAkgGA1UEBhMCSW5tYm9yLmNldDoGA1UECgkMCVQwHgYDVQQ
EiN0ZXN0YWJsZSBQYXZlbG9yZWRMaDDk0OzAkBgNVBAMTB3NvZnlhZ2dsZSBD
J arguing the domain of securitization
-----END CERTIFICATE-----

[Page 63]
Robert’s private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA09wUaWtxCSwLD52VCMFKIGbvn5wWbBa56xQW2hnMQU2xz/hT
W6UNduyXuhYqn+1XfttS5s+P0J80OFac+iMTk7zXYR5EgaBj0ju0DwM+oADWybnn
V44hkgqN6LSFESnBuVto3Nbj0BpySyYP9SY+R+FAqycxxf00E26ZpwwvdPCgtK
2OQC6ltQrziY1IEwefEOncOpA7c0fsfjzAktqf2/2n07Lp/S72bRIwNh+Jm0S9Q0
5x+e1fjHJdkw3NDq8Zzw295ghA5c19569FRk0gRzPFgg2Scdm/BZ9xQCeO25fN7W7
pq1Adg24zqQruyzu2UYwhvAzyWdCvqJQCM75jIwIDAQABAoIBAAv+Q3GMUPYraHbz
1th+Ekw86MFcUb2n8T9rjbeFjC8J0Qa/CgkAGPkp7FzBFwNyr8TXJhEAUHw+WzB
4PpHywnoUjfqFP8RavfmvVYNS1dssBtWytD0a41mWnDnF7vec99U71K5j2HN
r8NPR7et8a00xdFaY9G46WDkC0nkHBgMMyY/Vu2KpH0f0lhTpfLmxtS7We+zd3uQ
mva15Guc+8EL079ufpokchr4E0036c4u1CnqQfOUAkoXCMK7G5ue620IXLE
Cqeyp2ZPen8whSNG198I1CF15AeBo0ApMcMwrfcbnpQMHOQyQmh2XveqFg0GQlN
UAa16NEcgyEA9TrFg3Kuw1Fv+iKxtX61Jm0W7YgN443ntB/9+xSKcOicZ6loPoBT
VH5VqHhpjicucBuuya77Kr6I1Ahv7AV0sZFRHAb3M7wOvYgKtS2+12of4F6EMU42G
ISAcsS4cVFhYq10hC91B1YPXxurp0ybl1RkEeALH6anArEAeWcCqGHYAE3A9od
gEcaheQenu5P8U5Yj9y9fbABrQvxdQKn0w02trfKlgyKvgtnve7831EGojVHg23hr5Isk
IpxFgBiqeVEGuPv3D0Jc5zZTE7OweWBLebC/tC2znHBCnCnx8jxW5m/CtZmHuxV
VJ1WpUDn+K+7+G8IK0+Kp5QDdCwxXptHRKGPBCucYGAVgCulFLB33Vdqf6IpK10
TZEzap5bdyj721IFzPunuJyggP+tOnr87Taffip0gjr5g1TWLS8BNTZeYrQSr
iuqWP39EZeXVUa3z0RPNobi1Rajwrl0jx0406n4m317xeweje/J9IC9SOLqSjWN1
2f+ntWppM8dGerF6/Sdf+LQKBqCydaf2KeF/cHCmiXuHxXVhrs4kccTGof75Rdi
hQNdyp2ZHiVU9srnTivn2J5MPGJGskcFtQvpk31lqyShgkV43H1iTnB/A5p5bb
//7muexQruak59MKEL0JdNbcBFk/ffH26UWG7pE5cK/fHyVgQj93Uv07PBCAlYy
CunFQkBj95Wk5dSO4n4wVE1i9ia+14lng8pwJ1DUxnOCdV1dGzA2CnWt9
wP0r+jvhwK6V6Xi1m0tqcc1Z07pC3CJBEtAckH2j1k+ZAEqJnMf+6H2Rcv6Sbozq0
5dFCEBZwle1Z0qsmg3j8+oyiLzIUs/uzFkJ0JoJrP+OtpPKsrFr+/Y
-----END RSA PRIVATE KEY-----
```

Appendix C.  Message Dumps

This section contains a base64-encoded, gzipped, compressed tar file of various Cryptographic Message Syntax (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" would recover the CMS messages and allow them to be used as test vectors.

```
--- BEGIN MESSAGE ARCHIVE ---
H4sIAIpaUE0CA+ybeUTax7hHSClCH1poqSIQvFECu5tsDhAEDAQTbqsCQEXT2
JBTiyGUSIERE9Uii1R2qgyEYRWHucqK1UBWU1vsXJcexeI3LpfCaraUPuSFfj
TXJH/JPdm3f7TyZ8/nzfr8J76LEKSVCCVgTkgMd+yHk/0LAAAEGa0h8BEki
wp98hNsIQAxCIaHdbACGIRDCA1CBQCTqYAGdv6HEFWQ1stsVrkISD9zXVtvj
Df8++HcyZor+Bdgs0XvIm0U0f8HsITRMNdUjyjRtRqWb4BtpAJyNoy
Vlg4/37mxBwTgAuP2IhNyB9FmEMAA24CKTk3I1LUVKJoBo57yH9MggkaHUJi
CxAgsVAc3kwDqGQbu2YXhYvmULNCaIMgFQAdUeo8Zyo4RMF0MnNJ2hqm
zB7kqvu+u2n8FBiJll+1CSXqAAXh83aeTOlmoK1XkXKISAgEH1mpCvYgYjH0
l+qMYui49gsdpw/ky9M33V2MOAWTDdcPQ6eFSgusopp0yBZIrUaj9rZg
```
RFC 6216  SIP Secure Call Flows               April 2011

Jennings, et al.              Informational                    [Page 65]
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
Columbia University
1214 Amsterdam Avenue
MC 0401
New York, NY 10027
USA

EMail: kumiko@cs.columbia.edu

Robert Sparks
Tekelec
17210 Campbell Road
Suite 250
Dallas, TX 75252
USA

EMail: Robert.Sparks@tekelec.com

Brian Hibbard (editor)
Tekelec
17210 Campbell Road
Suite 250
Dallas, TX 75252
USA

EMail: Brian.Hibbard@tekelec.com