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:e4: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):
      79:e4:30:9e:98:d0:ec:07:b7:bd:77:d1:f5:5b:
      f8:34:41:70:d9:c0:03:91:6a:ba:d1:11:8f:ac:12:
      5b:c2:de:0b:26:65:d0:91:c7:70:4b:c7:0a:4a:bf:
      2c:78:ec:a5:0f:be:9c:10:f8:c0:0b:0d:73:99:9e:
      29:c3
Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Key Identifier:
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 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:l= 949 cons: SEQUENCE
 4:l= 669 cons: SEQUENCE
 8:l= 3 cons: cont [ 0 ]
10:l= 1 prim: INTEGER :02
13:l= 9 prim: INTEGER :96A384174EEF8A4C
24:l= 13 cons: SEQUENCE
```
26:1=  9 prim:  OBJECT :sha1WithRSAEncryption
37:1=  0 prim:  NULL
39:1= 112 cons:  SEQUENCE
41:1=  11 cons:  SET
43:1=  9 cons:  SEQUENCE
45:1=  3 prim:  OBJECT :countryName
50:1=  2 prim:  PRINTABLESTRING :US
54:1= 19 cons:  SET
56:1= 17 cons:  SEQUENCE
58:1=  3 prim:  OBJECT :stateOrProvinceName
43 61 6c 69 66 6f 72 6e-69 61                     California
75:1= 17 cons:  SET
77:1= 15 cons:  SEQUENCE
79:1=  3 prim:  OBJECT :localityName
84:1=  8 prim:  UTF8STRING
53 61 6e 20 4a 6f 73 65-                          San Jose
94:1= 14 cons:  SET
96:1= 12 cons:  SEQUENCE
98:1=  3 prim:  OBJECT :organizationName
103:1=  5 prim:  UTF8STRING
73 69 70 69 74 sipit
110:1= 41 cons:  SET
112:1= 39 cons:  SEQUENCE
114:1=  3 prim:  OBJECT :organizationalUnitName
119:1= 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
153:1= 32 cons:  SEQUENCE
155:1= 13 prim:  UTCTIME :110127183605Z
170:1= 15 prim:  GENERALIZEDTIME :21110103183605Z
187:1= 112 cons:  SEQUENCE
189:1= 11 cons:  SET
191:1=  9 cons:  SEQUENCE
193:1=  3 prim:  OBJECT :countryName
198:1=  2 prim:  PRINTABLESTRING :US
202:1= 19 cons:  SET
204:1= 17 cons:  SEQUENCE
206:1=  3 prim:  OBJECT :stateOrProvinceName
211:1= 10 prim:  UTF8STRING
43 61 6c 69 66 6f 72 6e-69 61                     California
223:1= 17 cons:  SET
225:1= 15 cons:  SEQUENCE
227:1=  3 prim:  OBJECT :localityName
232:1=  8 prim:  UTF8STRING
53 61 6e 20 4a 6f 73 65-                          San Jose
242:1= 14 cons:  SET
244:1= 12 cons:  SEQUENCE
246:l= 3 prim: OBJECT :organizationName
73 69 70 69 74 sip

251:l= 5 prim: UTF8STRING
73 69 70 69 74 sip

258:l= 41 cons: SET

260:l= 39 cons: SEQUENCE

262:l= 3 prim: OBJECT :organizationalUnitName
267:l= 27 cons: SEQUENCE

268:l= 290 cons: SEQUENCE

270:l= 13 prim: OBJECT :rsaEncryption

318:l= 0 prim: NULL

320:l= 271 prim: BIT STRING

321:l= 00 30 82 01 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 .....i,0,P,.1+Y

364:l= 6b 6f 1f 55 a7 3d-fd 70 d2 56 db 14 99 17 ...do.U.=.p.V....

368:l= 92 70 ac 26 f8 34 41 70-d9 c0 03 91 6a ba d1 11 .p.&.4Ap.....]

369:l= 8f ac 12 31 de b9 19 70-8d 5d a7 7d 8b 19 cc 40 ...1...p.}...@

371:l= 3f ae ff de 1f db 94 b3-46 77 6c ae ae ff 3e d6 ?........Fw....>

382:l= 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

386:l= 34 10 ab 27 1f 1c f8 30-3d b0 9b ca a2 81 72 4c 4.'0=........rl

387:l= bd 60 fe f7 21 fe 0b db-0b db e9 5b 01 36 d4 28 .'........[.6.

391:l= 15 6b 79 eb d0 91 1b 21-59 b8 0e aa bf d5 b1 6c .ky.......Y.....

392:l= 70 37 a3 3f a5 7d 0e 95-46 f6 f6 58 67 83 75 42 p7.?..F.Xg.uB

393:l= 37 18 0b a4 41 39 b2 2f-6c 80 2c 78 ec a5 0f be 7...A9./l.,x....

394:l= 9c 10 f8 c0 0b 0d 73 99-9e 0d d7 97 50 cb cc 45 ......s..P..E

395:l= 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

606:l= 04 14 95 45 7e 5f 2b ea-6a 65 98 12 91 04 f3 63 c7 ..E_+.e.....c.

68 9a 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

637:l= 30 16 80 14 95 45 7e 5f-2b ea 65 98 12 91 04 f3 0....E_+.e.....

663:l= 12 cons: SEQUENCE

665:l= 3 prim: OBJECT :X509v3 Basic Constraints

670:l= 5 prim: OCTET STRING

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:

---

Jennings, et al. Informational [Page 8]
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 (EUK) 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:a3:3c:59:0c:e9:bc:e4:ec:d3:9e:fb:99:02:ec:
05:61:0b:0a:ca:ca:ec:51:ec:53:6e:3e:3b:2b:00:80:
RFC 6216                        SIP Secure Call Flows                        April 2011

b0:2f:f8:da:c7:3c:01:dc:cb:2d:31:8c:6c:c6:5:
f2:d5
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

15:0e:e3:25:91:00:0e:90:db:d8:07:11:90:81:01:3a:48:a8:
75:bf:6f:09

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_WITH_RC4_128_SHA
  TLS_RSA_WITH_RC4_128_MD5
  TLS_ECDHE_RSA_WITH_RC4_128_SHA
  TLS_RSA_WITH_RC4_128_SHA
  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_DSS_EXPORT_WITH_DES40_CBC_SHA
  TLS_DHE_DSS_EXPORT_WITH_DES_CBC_SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5

compression methods
NULL

1 2 0.0012 (0.0007)  S>CV3.1(48)  Handshake
ServerHello
Version 3.1
random[32]=
4c 09 5b a7 30 87 74 c7 16 98 24 d5 af 35 17 a7
ef c3 78 0c 94 d4 94 d2 7b a6 3f 40 04 25 f6 e0
session_id[0]=
cipherSuite    TLS_RSA_WITH_AES_256_CBC_SHA
compressionMethod  NULL

1 3 0.0012 (0.0000)  S>CV3.1(1858)  Handshake
Certificate
1 4 0.0012 (0.0000)  S>CV3.1(14)  Handshake
CertificateRequest
certificate_types  rsa_sign
certificate_types  dss_sign
certificate_types  unknown value
ServerHelloDone

1 5 0.0043 (0.0031)  C>SV3.1(7)  Handshake
Certificate
1 6 0.0043 (0.0000)  C>SV3.1(262)  Handshake
ClientKeyExchange
1 7 0.0043 (0.0000)  C>SV3.1(1)  ChangeCipherSpec
1 8 0.0043 (0.0000)  C>SV3.1(48)  Handshake
1 9 0.0129 (0.0085)  S>CV3.1(170)  Handshake
1 10 0.0129 (0.0000)  S>CV3.1(1)  ChangeCipherSpec
1 11 0.0129 (0.0000)  S>CV3.1(48)  Handshake
1 12 0.0134 (0.0005)  C>SV3.1(32)  application_data
1 13 0.0134 (0.0000)  C>SV3.1(496)  application_data
1 14 0.2150 (0.0154)  S>CV3.1(32)  application_data
1 15 0.2150 (0.0000)  S>CV3.1(336)  application_data
1 16 12.2304 (12.0154)  S>CV3.1(32)  Alert
1 17 12.2321 (0.0011)  C>SV3.1(32)  Alert

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.
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
Max-Forwards: 70
To: <sip:fluffy@example.com>;tag=ef6bad5e
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
--3b515e121b43a911
Content-Type: text/plain
Content-Transfer-Encoding: binary
Hello!
--3b515e121b43a911
Content-Type: application/pkcs7-signature;name=smime.p7s
Content-Disposition: attachment;handling=required;
                      filename=smime.p7s
--3b515e121b43a911--

**************
* 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)
:        }
52 420:      SET {
56 416:        SEQUENCE {
60 1:          INTEGER 1
63 125:        SEQUENCE {
65 112:          SEQUENCE {
67 11:            SET {
69 9:              SEQUENCE {
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 {
138 39:      SEQUENCE {
140 3:        OBJECT IDENTIFIER 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 rsaEncryption (1 2 840 113549 1 1 1)
214 0:        NULL
216 256:    OCTET STRING
240 256:      74 4D 21 39 D6 E2 E2 2C 30 5A AA BC 4E 60 8D 69
242 256:      A7 E5 79 50 1A B1 7D 4A D3 C1 03 9F 19 7D A2 76
244 256:      97 B3 CE 30 CD 62 4B 96 20 35 DB C1 64 D9 33 92
246 256:      96 CD 28 03 98 6E 2C 0C F6 8D 93 40 F2 88 DA 29
248 256:      AD 0B C2 0E F9 D3 6A 95 2C 79 6E C2 3D 62 E6 54
250 256:      A9 1B AC 66 DB 16 B7 44 6C 03 1B 71 9C EE C9 EC
252 256:      4D 93 B1 CF F5 17 79 C5 C8 BA 2F A8 06 DC C7
254 256:      62 A3 F3 1A 1B 24 E4 40 66 3C 4F 87 86 BF 09 6A
256 256:      7A 43 60 2B FC D8 3D 2B 57 17 CB 81 03 2A 56 69
258 256:      81 82 FA 78 DE D2 3A 2F FA A3 C5 EA 8B E8 0C 36
260 256:      1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50
262 256:      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.
SEQUENCE {
   LOCALITYNAME (2 5 4 7)
   UTF8STRING 'San Jose'
}

SET {
   SET {
      OBJECT IDENTIFIER
      ORGANIZATIONNAME (2 5 4 10)
      UTF8STRING 'sipit'
   }
   OBJECT IDENTIFIER
   ORGANIZATIONALUNITNAME (2 5 4 11)
   UTF8STRING 'Sipit Test Certificate Authority'
}

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

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

SEQUENCE {
   OBJECT IDENTIFIER
   RSAENCRYPTION (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 F8 28 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 C8 BA 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 E8 OC 36
   1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50
   9A 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 4E 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.

```
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 {
41    1:           INTEGER 0
43   112:           SEQUENCE {
45    3:             OBJECT IDENTIFIER countryName (2 5 4 6)
50    2:             PrintableString ’US’
54    19:             }
56   17:           SEQUENCE {
58    3:             OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
63    10:             UTF8String ’California’
75    17:           }
77   15:         SEQUENCE {
79    3:             OBJECT IDENTIFIER localityName (2 5 4 7)
84    8:             UTF8String ’San Jose’
94    14:           }
96   12:       SEQUENCE {
98    3:         OBJECT IDENTIFIER organizationName (2 5 4 10)
103    5:         UTF8String ’sipit’
110   41:         }
112  39:     SEQUENCE {
114    3:       OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
119    32:       UTF8String ’Sipit Test Certificate Authority’
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

4.3. MESSAGE Request with Encrypted and Signed Body

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: MjYwMzdjYT3YWRkYzgzMjU0MG14Mzc2Njk1YzJlNzE.
CSeq: 5449 MESSAGE
<a llOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
    application/sdp, multipart/alternative
</a llOneLine>
<a llOneLine>
Content-Type: multipart/signed;boundary=e8df61ce5d1e864;
    micalg=sha1;protocol="application/pkcs7-signature"
</a llOneLine>
Content-Length: 1455
--e8df61ce5d1e864
<a llOneLine>
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
    name=smime.p7m
</a llOneLine>
<a llOneLine>
Content-Disposition: attachment;handling=required;
    filename=smime.p7
</a llOneLine>
Content-Transfer-Encoding: binary

***************
* BINARY BLOB 3 *
***************
--e8df61ce5d1e864
<a llOneLine>
Content-Type: application/pkcs7-signature;name=smime.p7s
</a llOneLine>
<a llOneLine>
Content-Disposition: attachment;handling=required;
    filename=smime.p7s
</a llOneLine>
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'
:     :     }
:     :     }
:     :     }
:     :     }
:     :     }
:     :     }
:     :     }
:     :     }
54 19:       SET {
56 17:         SEQUENCE {
58  3:           OBJECT IDENTIFIER
:             stateOrProvinceName (2 5 4 8)
63 10:           UTF8String 'California'
:     :     }
:     :     }
:     :     }
:     :     }
75 17:       SET {
77 15:         SEQUENCE {
79  3:           OBJECT IDENTIFIER localityName (2 5 4 7)
84  8:             UTF8String 'San Jose'
:     :     }
:     :     }
:     :     }
94 14:       SET {
96 12:         SEQUENCE {
98  3:           OBJECT IDENTIFIER
:             organizationName (2 5 4 10)
103  5:             UTF8String 'sipit'
:     :     }
:     :     }
:     :     }
110 41:       SET {
112 39:         SEQUENCE {
114  3:           OBJECT IDENTIFIER
:             organizationalUnitName (2 5 4 11)
119  32:             UTF8String 'Sipit Test Certificate Authority'
:     :     }
:     :     }
:     :     }
```
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 OB 11 52 A9 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
: 28 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 4C 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 OB 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)
44  1:       }
52 420:     SET {
56 416:      SEQUENCE {
60  1:        INTEGER 1
63 125:      SEQUENCE {
65 112:        SEQUENCE {
67 11:         SET {
69  9:           SEQUENCE {
71  3:             OBJECT IDENTIFIER countryName (2 5 4 6)
76  2:             PrintableString 'US'
79  4:           }
80 19:         SET {
82 17:          SEQUENCE {
84  3:            OBJECT IDENTIFIER
88  6:            stateOrProvinceName (2 5 4 8)
89 10:           UTF8String 'California'
94  8:            }
101 17:         SET {
103 15:          SEQUENCE {
105  3:            OBJECT IDENTIFIER
108  7:            localityName (2 5 4 7)
110  8:            UTF8String 'San Jose'
115  6:            }
120 14:         SET {
122 12:          SEQUENCE {
124  3:            OBJECT IDENTIFIER
127  9:            organizationName (2 5 4 10)
128  5:            UTF8String 'sipit'
133  4:          }
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 done with SSLDump from http://www.rtfm.com/ssldump. The book "SSL 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


Jennings, et al.  Informational  [Page 33]
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

#!/bin/sh
set -x
rm -rf demoCA
mkdir demoCA
demoCA/certs
demoCA/crl
demoCA/newcerts
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 is 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
[ sipuser_noeku_cert ]
subjectAltName=\${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment

EOF

cat > demoCA/private/cakey.pem <<EOF
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIDdjABgkgkhkiG9w0BBQwMzbAbgkgkhkiG9w0BBQwDgQIwlwc771D1NUCAggA
MBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8w
IEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAM
R8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCw
UAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQE
BCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3
DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqG
SIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEG
CSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwM
CEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMD
kwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTks
uMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR0j
TksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQYDVR
0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMwNQY
DVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GCDMw
NQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0GD
wNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBHMA0G
CDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQMBH
MA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSIb3DQ
MBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCqGSI
b3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQGCCq
GSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQC
qGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgMBQ
GCCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEgM
BGQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3DQMBHMA0GCDMwNQYDVR0jTksuMDkwMCEGCSqGSIb3DQEBCwUAMR8wIEg
MBQCqGSIb3
-----END ENCRYPTED PRIVATE KEY-----
EOF
# un-comment the following lines to generate your own key pair

```
# openssl req -newkey rsa:2048 -passin pass:password \
    -passout pass:password -set_serial 0x96a384174eef8a4c \n    -keyout demoCA/private/cakey.pem \
    -out demoCA/cacert.pem -days 36500 -config ${CONF} <<EOF
# US
# California
# San Jose
# sipit

EOF
```

either randomly generate a serial number, or set it manually

```bash
# openssl req -newkey rsa:2048 -passin pass:password \
    -passout pass:password -set_serial 0x96a384174ef8a4c \n    -keyout demoCA/private/cakey.pem \
    -out demoCA/cacert.pem -days 36500 -config ${CONF} <<EOF
# US
# California
# San Jose
# sipit

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

# 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 "$1\n" if (/\w+@\w+\..*$/)}'`
ADDR=$1
echo "making cert for $DOMAIN $ADDR"
if [ $2 == "user" ]; then
    CNVALUE=$USER
else
    CNVALUE=$DOMAIN
fi

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

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

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

export ALTNAME

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 `${ADDR}_cert.pem` \
  -inkey `${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.

**B.1.  Certificates Using EKU**

These certificates make use of the EKU specification described in [RFC5924].
Fluffy’s user certificate for example.com:

-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pNMA0GCSqGSIb3DQEBBQUAMHAXCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDADpYWxpZm9ybmlhMREwDwYDVQQHDADYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IMBIGA1UdDgQWBBSFlwm401U3JIrc3uORcuQiz5iHUJafBgc7
mBKRBNjx2iA0WZXjiZaLBgNVHQ88MA0GBWcCAYIKwYBYh43wQxIzj7MzOoEy
LKrjIuacV79IKPd2dBP88jwmmfHByKe0+RK43W9xTDjeBg0zRQvL5nis31dHv0
mmiuiuLB0btC281wyDcENxOiroF9Tix11RnVvETzy/8i4P8OcBglmDzLCN8Mfa
TrHzcFTPbDhtRlmH2Rbsr6/h/EhMjHgrb9bX/XasVDuMlkQAoXNYXBoGQqE6SKiq
nrBi0zBwLCvpxSUjYPfArWZIn3C3RuqjzuRzegeyG4GyUcLvZ檣HIs0NUBnLgH4
we+5SSJGmE4acGONrmo/v1MAyyg41Ltwv4RLB4GRO4fvpqdRLS3V1v28J
-----END CERTIFICATE-----
Fluffy’s private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAoyxZDOm850zTnvuZAuyxNjq30x1Nwzq2rlc9X1UI4d4x+pOnw
aDEojMyVnhdPnenbf2nIioBta9hvV9+wRTlmCmONE4l1buDrzbJ7141qysBHgy
mMJ152q3LbM84+uPxe+LWUQJF3v+p4GvN6fn4x+wjdbLyL2wQqsYBaOGP/VZNfc8W
3KoFYA4xysrsUeXb70rAID+NRsGCmEtiAtE88z9Kw60ogug14QULu4rd4y/BGp6G
mhqokkXk+BJtNWDb2wl/jaxzwB3MstMXsxi1y0Y+iyoA8r634BR1tix1H+GoV
i/slnhNERjTXJBzPVo3UvtaUW/Cm2+Pdz7Ty1Q1DAQABAoIBAH+b5vijQjr1WnnW
YM78s4mpWeDr5chrvmMQsyu/zQe11u45sL1F9gcOl1DQGtpFJLa7z5Ug4nGYjvQ
3QG61eL5mkfddDH2R+z1s3WuMmYQQ22TaZ41VWdo+V/v8Ap+T9y9A2UgiwQSoA/3
R0PLN31Taws8nE+hwiaG5sweujBvcaIJu4ROrqGHRHaeEplU+tfjcHHE1fzUAmKyM
mcQgFBIpducAlpHyHe3Pyc0oGnLyEVnv291xQWQifWT7nfq7K0QDLa6+TvbG3fGEY1w
WK4DMraUbZ66Jlnj1xFADoxW0TsygV+KYhZcbwjBWAUSOduAtfwa6b72OnWd2B8J
8KYrVXCegYEAeCJJZGavifiqxsCWd/WQ8S3SimI62KSlrN3b10/R0/60KUI2a3p
162hNHqBt3DjpkWiZrux6s2odsU7k3z6q+qm++POTUwL7z3Bri0FmquV5YgA5
ZmFgG7wLAM29zhv0H2tjGrrwM1NSyJt2t+jyqipi01XqkbdBpPBxKPrdcGqEYAw09f
4M2QQBfzjrecPeQpwJqnh8cuoHS+2CNYjGjmjd/zAUgVF2+FPA1R1DmJagJ9jihw
15X3CbnknpKbfhf1mKcCyG+A+fjQaisq/NzN3yA0FP9Waat0PoBsAhT9X5FwXH6
YBKUqrgPFP5DAy427ElnsIRa+LtoPAdqpphFzMCgYEA1gSO0s2FA43uyTpeF3t
rmQpVilab7KFsaigGBgUY7p0kFO9rDwVT419sd48a7kb09r2K08sHe2z8BenoB
Oj+HijNHYHSTxrJnQbLZqTP2fMUuPDEFX/92n6WFjKBX+d1PSVJ8xUKiiJg3c6/H
1uhMzQZFbKXXVOPTRG3G3DcGcYEAoFPmq8QZOIA+BbnqziV8OsfuN8geFye9JRm
55jPkd0Hz2Xts3dTmMBzGI5KUb9nbV1Gb/PVBbcoST6vtD0kpyq70a9aAcyc
Zv5SFAnFy0vt9NAcsHixD2C1dx7UEja4PQh49NsaK9aGD78yFPOqUUsvp
0i0XmtjCyEAxUiku1+kSwIrnc1Ut0gt6+4Tzczj5E0eEpRTktz/IsnNEXeA6N
EUqWLMoNChp72V5tvxKkgxjO8Vpg1ZehHIt5jzb8X8MmBisQXvTF6rps38Fq1M
EtXf7nTdzJutR7d0g2uG4boJMF590nqujrj9VeSxEWURSk3Yg/h8=
-----END RSA PRIVATE KEY-----
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pOMA0GCSqGSIb3DQEDEQQUECwGgAwIBAgIJAjajh
BAYTA1VTMRmWEQYDVQQIDApDYWxpz2m9ybm1hMREwDwYDVQQHDAHYW4gSm9zLTEOM
AwaGA1UECgwFct21waXQxKTAnBgNVBAgMBWFwQCA1K1GjAQYDVQQHDAJXcmVzaGVy
MQYDVQYDVQQGEwJUZERDTATAKCIxKmQwMjA0NTE2MiwgBDACBQAwMDA8QjEw
FQYDVQQDDDAJeihwGAYDVQQDDBNmYzEwCAYDVQYDVQQKDBM2N1d0d0
QXV0aG9ya28xMCMnNDInNzE5MzIxN1oYDzIxMTEwMTE0MTkzMiE3MjBwM4sw
QVYDVQQGEwJUZERDTATAKCIxKmQwMjA0NTE2MiwgBDACBQAwMDA8QjEw
FQYDVQQDDDAJeihwGAYDVQQDDBNmYzEwCAYDVQYDVQQKDBM2N1d0d0
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAy+aHVXQN4BX/zliKmzGPcpe40F3H3sVpKV206bCzcX9g/IC
1F/Pz1J911E2z9NF9Q6gPjJD1I+DTRqQ2cncqCK3Kr11Fyu1EJuaCskOlWtdQ4oxO
6nT01shK0YW0VIT499FxrrQBT2EURvyGX0UHB/AMNcpF0qeteJtZmPV8C+S7hzYp
gI1wAqxK0PaN5g58NLYVPF1lldVBTUgNZQ91iW/VW/86i4oN06rR6H71mqsKAR
ud05VRuW3FdpqcGR5whaRXg0w2BNT9fe14akbSKq8ca+uEaPQ5z9JmBnt5Ub
3gGYXSiGwh4BmiqIK+/kmQ5y3qsig6xUjvw5WwIDAQABAoIBAHCXmrGgRS0XWLW
PLKm+1LSrG14+aqwbq663SHTAB1Yzvu+W2Bo20MnveJrEeo4Q72J6boJozZvF
CKmKqrYiKaXgRBW/jtZ6xCWGCNCAL1pnX1lWG5tDIg8SALOO4N7hyR0rrA4Rz
W0vuVQSYFXX4BhvdxZesyRwCqn3x0pF8ff95Ad+vuJd5CYuFZCuyGkszQ3fi+Nia
Gqs01Euoi1Ev72rsw2E5+wtx3qX8B24HXR+yQ9NhbE81p2CwId1UhqlIH18kwWmM
G3oLki7owV+M6zX/uwsAMF0rmdn5kET+b5D01Ik5uAa8LZsf95oVkJgw7aZaj5e
xKhAdGECgYE8930YqU2+AcEkjC5hygw1M/X5/k/Icv2p0a8/in2hJW7izGh0AF
jjxuoIVxbsf9cZ+cM6g76sww9emcvolArqbbhFafVbZCsrLeEahQtGcu3wV06vX
N0EbbF5Fm0K7qaQ1Sgqj0NF5zP2JsrxGNoRmgFwFwVdcp/P/3Jp/I1EsQgYEa1gu
/71h89ogldmTPzmpvpnAnA4r/luMX9AE4LNRp09Jx0B7Vuat1ABtx09/ZN1hLh
BZTZ2R52RjzSHXZ3fdoMgSx9Q3q+a+xuPe14RcppHNjdYkPdhPn0UwQBqFL6kyU
nTEF+k6VIzVnSmGbB6wpHU1cjDAZux71p6W49TEcgyAHMha7PExUDT076rH9tpCe
sume5441lshtX0W8oAIpVcuqzeRdKmBWB1B7WYoUS3Yq9H82JoPM81amqfwQuMzYh
/5Y1AIw1Jk+Q9VnZJNm6ohDvFVQm9VCEH1S/Mmox6Fw8EZLJ7HvAzzzy
Dqhtbh6wFW5WYM15z3xewKBqQCRm1Ky/QQFm0+Ih5ZMgB3e17GGLB1sNe0nY1Ve
Dzv0pc3UJqHQQ17CLUdUy9L19V988t17+V76JXlHdyY9U4bdBau/kkgQn+gd9PJ
Ul1xq8aam73rUJLXW7H68A16jQmi4tpcWN5S/pr51n0Uy1/hXT7psFZAx08
Ov1lQK8KVbQGdAzCYC/6wumGJuserVczZd/H6+E3ntZmz273c8+wV98r6RZzUoJY
bVnRyFz91KFxlNGRECZU2V2DXHUAguge05rbzPudAZ4wsrncHu0yw8LKfXHdkct
pVLs0vHRK2W/W2I+t2exvSPQPt30y8T6fsB92Nb/H4D160heHkuQ==
-----END RSA PRIVATE KEY-----
Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pPMA0GCSqGSIb3DQEBBQUHAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZmluaXR5MTE0MTE0MTKzMjE3WjBbMQsw
CQYDVQQGEwJvUzETMBEGA1UECBMgMDA5MQ4wDQYJKoZIhvcNAQEFBQADggEBAl
-----END CERTIFICATE-----
Private key for domain certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3XQGAhDC5wQfviy2JOeb1KNIN4WebYMShFAajkix+oaMp4C5v1LspspjR4St9nSFghZ+Tj2ACmQsIK1qDmp/Nc9wcvWN96UNnqUzgR63TK1MCshDSpzb20djXW/S0S5mdqffqCnpaK+SREBiB2qKEjcfSFL5GGVOwT8EFDvgEW6XoSXdsEgjcGSHYK91ViZFrnaaJ1TGfTCN1ht0WVLa3R1IZK5DU6fHfYD WynKsM3gAUoHbB50ArpjWVzheyujB647aObaeeDD3C1h6bLjs3HiPs/ar+z6zv+XvuxcX1028oEdUCyuxbVoaL1c0xm/xP2EaJEn3f2FpSxDAQABAOIAB9s323ni4Dk40wM u7w48acCFILslSMLzqoMEKwCN6FO4zDTc23LagaJxje0UMuuKXVFEYWA6r6BcTM hYQLQMoOCdLNX4y+d+2tUJeLq+9aAUUu093ebDxcMntkfh6yNyUS/mk/KQMbFRT1dDr0wXxSjc19I6yxArkB7/9UEcDut6vzdbz+aqXSpHZH47je5O2WZXKHzsYobM8Y8c2XwudP1zdqt0rOrjrexxqOQf4CBqBxoGmbae9Wf27Kw2bBm5+blZFdqNxoh 6Q3rJ9EDYWkrVMAq9a67a59lwSTl1myC0c6FmFoCMG1g0MFHcEdvnuNYPwD322oK ZdfsawECgYEAAeWMTdhAE+9T1d2qi1LQV+y8bdTHQ9rSgQ9SF+q5SbOpqa79ER asuDuqxU+Tiews0ircrKayQcCl1fnfB7h5y6GupUK8HDLLkA29FVZJe+Y42blb 4TEy/RxEECQREgtneQwio801T1dobNwxzVsi3mhrtpofpPBBERZUSsCqYE5AJG2 aGRckyzASGAnZmqXCP/p1mU+tJb2OCqO6/3gsxi/191LwtRhfGpx/pYCg2Wlpbz +mpnDqexKtow1dbjorrUADw84Zg4u9d+uWOCCExPCVIEu4DZsRURDy3ozpkL1vJauN MLqB1dJkUFRxTI4Rzx1Ysfs6ndWAXdPDdI+GEcGyeAoyFrYy+dohSvs9uiyj4e FV5n5t8E7lQF7L72SoOdLHy1JO1V2+VF71erbDusJ751q9hj1q7id3ips/M87P 2qJSMTgBoJrSTs0LML6DSFmMy/jFY1beaMZ96nPg34ip38RSyPrThnV7kpo3 Ao7AtxtwVzBPVuc78A/8ECgYEAw2ps2F13qdql3ns01Ho3gqVoaGUUU10K2MI wjyMI/AK2r4PKthmlP1EPt/tpsB2yBB6XcoYaY5+10DWz0ygGHNiheyR7GgRh pqC0EHEquizkRd9hurz5gi1oXcCQ4tBV+wL7+yNKAUIdF3g7iG1ZUA6fjxew9io FzBxg6ECgYEAhVSequcmdotdPQg3Fk1CmtH72FfnF2rsuBA0nWhU690te+ +Bmd4fUB99tQ0UX9dernktor3+w1JKHEFjm/OFxtQQ19ogRe4e69j0wXNKsSa GjGUrzQ3Vm2baMM7sE8C5mQ9mskDzd1vAB2Bmp23oP16cvP1boE=
-----END RSA PRIVATE KEY-----

Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pQA0GCSqGSIb3DQEBBQUAMHAXczAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxp2m9ybm1hMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc21lwaXQxKTAnBgNVBAsMIFNpcGwCAQgGB1UdJQMAAwGB0GCSqGSIb
3DQEFC1oIzElMjIwNVzIwNzE5MzIxOjFoYzIzMTEwMTE0MTkzMi4wJjBbMQsw
CQYDVQQGEwJUVzETMBEGA1UECzMKQ2FsaW5vcmF0ZS5wZ2cm5pYTERMA8GA1UEBhM
BuI2U2F0eC02Ux0DAMBgNVBAoTC1BRIzAuMPAwggNiMB0GCSqGSIb3DQEBCwUAA4G
rMBAHAwHQYDVR0OBBYEFNiNYjKOu6f046JHy28GDRVME7sMB8GA1UdIwQx
MABaJGVFf8r6m6WYEpEE2SPhaJpYFhncmAsGALdnQEAuIF4DAdBgNVHREEIDAeggt
-----END CERTIFICATE-----
Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA7Cx2A9VI8zB4tawEMdRJclLPWTqm7qoxWDwhYx/WDRNZ9zm
NQBCVAsu22Y3IbxLOrnFV5y9/essyiYk91pZt6SINODUpmbXbCDI/6/H8d/UEHKP
ezYMT8qmgudCMVRbAkkcxs6QK1Jrpo0IVOYP2cdDlxxgNXE2YoYrP/7kdjF7oHIjo1kY
2ZIHawM9GtqtwDwpRyE7Tcom+sYW3wF20C0X/kLfx5hgamcpGWpk5XxaJQanpWdd3gu
u8NmcTc1D+j=3s4VuIrgfa+Hdy6H0sQxuT4+gw/BW/ujN6KXqpbK0QQTZGcOGF09g
xc8rJ6ddffwHnZ83K037aaxnxjHJ99NmUavHQ1DAQABoaJBAfBfByR2B1fp2f0186
yL6a6jWrlILhD76NFxrr/Alg/798648bEjCNIo2N5+ckxXv4x219NO0t+ttztiL3
KGyfKec06isncljxKgn0nz/03nOn197Xpxb9mL9t3GHOYRouK6xGpGILo60BtCz
F+8pk0jegc7eVFQpUMtUHmF/CmpyY30N5cvCrhAE/nCw/49b2mH3Gq+cmr5UcKfDk
baJyLd8Q1f+usMr6yYzzRT5c+4wmrBjujw9poMuEo4s1RaDnyeKJPSNR/6/LJk
tqnnQnI9c9j9wjF6hWA23dDmmU/kSRTn1KOz5zmV9Jbo4Fue4Fvnm/hj50g4CP9
h2UW1QfCgYEA+vNP2zpSgFS7jSebVvNyjChvqJ0nJAlSkMioRQTJ5PRm+Qo6GU
PvN4E0Ms8T7JyxvkpoaQ36Vlw/Wr0juM+n+dv1T1IFWTas8RNmzhHmOvlvFees
Qu2F1T612d/L9GBMCY/Y/sucX589g+3LC+Q09j9w8ehWJZ2sWYER4dsCgYEA8WXY
AqDkgjHRqu2h248gZsuogiZq05iuXhkh2VTQo1m92mu8m1htkaeov3/wojquw
TAQzb/t8EfQ7LIGJaKqaAua7mgG/aNB6MGpwdpBAFUZDL+DulKfDzBzTOL/IuaW0p
40RC0Up5nTUwzjK8b96nS550KXXxiUIphfCcCgYAk1EyDpmaiU0fxJ792rBuvV
82KwWbzwXqpxyLsvGzestSaRs45a9S2qiyLveIEl3m252i01iRSU/0rCnpPvEjc3g
+sY7Gzkhb146p3WuUbGmHMc0Nho43+3CM/JagGltmbwbWv7+MwtNT7iVH6u
EuUkU1biHxsV5zAbWeKhwKBgbBy5HwflCEXbA62o9NdhImPy28YuCLQR47kJreyu
MNz6A1Iqayah2iTxbGO8f9eAeDxxYXZkXfMK11EnFrwpf4803cMcM5nink1lVO
kwqQwOidrods3jyyazTzcC7jvXKKsOfjW+b2A9kz7Zj9v3HCg2qbaU5ttraio
JMMFAmqAH/B+k+e48w8wb+Xbb/9/DgAkSoK5vzzLHeWB9bE16Z6e4n/oFEGXVp
zqwYov6oZaqofmS6an6/6P27NC1QGJzxy27eB1Tlp1NCX1GitaQEtXmmtvnAznzXC
5Ue0Wv5bLx0nhbjWn8WzwJhYp0UK3pn99Gcf+vjk5Eg7zftg=
-----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+jCCAuKgAwIBAgIJAJajhBdO74pRMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTA1VTRMRwEQQYDVQQIDApDYWxpZ29ycALDQgMKA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQC6VyOIP6UANXy766KHiYDxyOpYFb0IoLJv6SeTw
QUQoZ3hQurnFidou4gCspblzaMotomy71nUexBFUKdbJOWGmcB2hrezJ+6rwJPK/
bF5YdiJtVgMrd51v/Ni5yzteH股份MszWzv33t+ojjak4XTjBmP2R00T67GUpEhrFv
sDeYTwi+GbldAR6bf6JDba2K6DnmkxTT5r6oYJH/IpYbubk28asBQN6EGBBgPEO
RReJYrj0JR/rBDe1lbx+KONFXP1wxJ/1TRMPmvvYUraWgTj18TtXISgfFhtaa/Y1K
YP79yun2N1L/QOcPic/C6CXbs3yAUK3qQO1G65pXH9KMK1NagMBAAGjga4wgasw
QUQyDV01REcswSYWc2lwOmZsdWzMeUBleGFTcGX1LmNvbVYVaw62mx1z25QGV4
YW1wbGUy29thdxcVzomZsdWzMeUBleGFTcGX1LmNvTAJBgNVHRMEAYjABMB0G
A1UdDgQQBB77TCT1Q5GWxwGZNY24mMVUuEnRDAfBnvHSMEGDAWgB5VRX5fK+U
mBKRBPXjx2iAWB3JzALBqNVH/8EBAMCBweAwDQYJKoZIhvcNAQEFBAQDgCBpB
wUGRhCQdhjy4bx5b5r5Kwz+NHvbsb8rj1Pqfcdnu3e+e/r+YuxOG3HwW+Mza
U2tUehwz87k6SgdqAdzl/CP2jmzCJo5uDhitzj6z6k1TSZyQrlSF9v/AgcUEF
9HU4Gkix/htaoEMy2z2Nz5OjdFME9w7wb3GxxqWTUz19TtoeCXYmLeQo/jwuw
40ybu1PS1IW05Md2Y5unFtcsRrZs1ILyTAvfANtLBrzFV+S87awrddeTT+iB7
Jjtt24U4ICMC8MtctH81PFPuRVC2kmhNEuQTuze1CslxKgY2+kn8Itndlv1mvLpXa2
2Y4ICPLCSj9AlqszL9I=
-----END CERTIFICATE-----
```
Fluffy’s private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAulcjiD+lADV8u+uih4mA8cjqwBBW6Cyb+khLcFFKLGUt4ULq
xYnTrulJARkW5c2jKLCu521HsWxcVCNhWlhlhJADoa3syfuq8C7Tv2xeW4o1bVa
jXEz2b/yZuc7sX6l1p897fqI4gpOF04wS2zj9kTtE+ux1kRGxVbA3mLVoYhX
mwwEem3+1XW2tiug55pMU+Ua+qGCRyAKWG7m5NvGrAudehBqGYDxUKU14K6xU
6wQW3tW8SvjjXRVZv5cIyF00TzKb1GK21oE4ydfLvEcBDybVwMVm2N5mD+/WLp9jzF
91EHDyHPwug1wbnN8gFct6KdtRuguaVx/SjDjTQIDAQAoIBA8tIBLI+8K5eJ1vw
/MoxOwKrMrwf8EftQppGTxhfjN31MbfIFAS5hJd3GncDqwAM1lyks6YEZ+mu/rnH
wp2FXCXO1FgSebd8tCMi1b027v0fXZUkTxR4aj41Y0HYRg7yfRXjjer8tWQ1KPMK
PVKmLOwpk34+2j0OhQUDpR3xhcJClQ81fc1hKe2JoixNDoPdfM3azTq8QUQPQD2I
mjww1IH1677G5o/6cMlOcOM0Feqv/3cUWIrmVPy4eyGHHdNtuFXKFpB4DQQMQL7TD8
FoDHBymHlOzsSf+gYGFOboYNgur2Cq2zrEFd9cf0RorrbXf6tM+akclxHhkhkKaa
JP2osbUCgYEA4MaetKsa7azhEYMc4TkoxxhV5hI61j1xR/6h++uYFOO1OBjM9yU3
5n6vLpygJhNhBw2bK08i1WPOF04syvvyKRY2elmUDraH29DKatRLEU9K82RG4AmXmK
G62sWOFx6JF3sONAvKj/7aN9jmK4K1v6EFyQGYEXsbP40fFbFbJBAe28CqYEA1Dnxm
ikJD+jw9y9Pkh51YJ3r+a5qPPNvmjGKKQo3ye36+rS1xmW0hX0CBOYrwhHBRA/
SxH93PZ8reCjNkhxp6AO87XZGco15UK6x/rwfd/3+sAhqPruGAwDl1ngkcu8VRp
8uP2CgjoDBi5U/YUR9?GK98x8kSf6kDT32mQMCgYBKH3R8VY7j01KqcTc1UWl
J1E3/gB4S+wQ8YElth0FVCP0sdSuZdILtfW070fUrea01k/SHeSIfijDihqN6zm
oDFMP+7vrb4z7UwrZPCq95n4k5ihTkNR1nV9elJTudjLcWS3pFC2JUI3OB+e+n
k6zuF0Cu0wFSCI2juibqwKBgCT6RHe1JjkJDe2FniX8r7D88y/w9wXVtDwqgiE4x
XQ/0F8A6IjBkTA3q5CpZ2BAVx9bIPjc7VEta21A8FVQPXVZC3sAAPXaha413E5v0
WYb1l1L17XZAXvJyv8W8/GhdN00zU1lTRG6w+5sA10r1/M8k/ToKnAsHfeEysEbo
y/W3oQASjoC9Fjy2aBV8HSSqaimX/R3oF4myGOWtHxrxXmezo4YDcMI0d9zR
a/AQrvVoFHRwyoak2kAIPrEgywCgDMs1h9xyAixFw23RfCq39D1y99TRrRkwa
ArJmcEdRESoSiyhXGFeQMG1yjUUXMWeYcLtqQkwI2LDfYFQ==
-----END RSA PRIVATE KEY-----
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTA1VTMRMwEQYDVQQIDApDYWxpz2m9ybmlhMRcWdWJIDQYDVQQIDApDYWxpz2m9ybmlh
MAwGA1UECgwFc21waXQxKTAnBgNVBAIBcHJvYmQwNzB4bG9tY28wYTEGCSqGSIb3DQEh
MAQIwMD4wDQYJKoZIhvcNAQELBQADggEBATvzoqPUE62p5xN9kX4Pz9L7L7C6q
s0pc2uelsMjwV5WvITq5s+D8en4aF2x05SnCN3I8s8pMnV6Ez5XzWVQz6jU4v
bTv32VZQ13lKzWhW6zv0cRf6Mf1vMRYQ9w7QTZkOw12W0q6v1
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEArPoF6eE5w7u1L+m9HJhyBR/dyoTW0xMgorIq4udN1AYqR/x
YEYDwV336yCFUHSJqC3iSwvR3bAVRxVr4PqUZSlRA4NHHDDBeQFW6tzCTjY5mu
SNJcKSSoEV2ENCv/6F//o2XmpIRBcvoFcmq9Cm1InSOeir3C3ERos/axyWufZR
4OrzSVdf16J23LO7lI2496mXw+6IqSOD6EJV1XcrqvkswFXujLFOzcC5qUKSM0
HUrCcsWglFvglMsbp4V1oj2FPPFDOL142nGbo/BryfMN144CyPU1lbPK1iJcpwBi
2ke4K0wZhpdHuvnIO4VrqBgZUANEIdIfMI1kM+QIDAQABoIBAduLR+kwp3sV1cX
Z34FsfomBALNeKpa4+KJ/JC7rXq9bfACXhec2AnuWnZ26TUNRFgoKl2DvookYE
gHD57n36dcf9KR7rphH5xi0OrL1JNcoIRFNeFpRNZiCZBuNiaAXFLnHGtznVnpwT7xI
axMnqsrU6epioO/ouAPkOu5x6e0+j+j3zau14EFd1w2R6moBMUvATAtuezEyLuC9A
6bFz2AFDchPvLwsJNmU0tAtJc8Fssx8KIls9HUXGS22eUFHwFkGWChwU60obGmas
E7GS7h4g9QvqV4qhGsvy9/MmQ88Gm70LynOyazFBCpuijQTHwsD674dM5l4kXYk
jcnTAkkCgYEAjbN2IIs3uWTvJtVnMrWn1QoZBDdh1LuNs5oi1xtO7C7ckcUvUs
nqCqOYozN/9N8Us12os3csXHyuuMrjwAVF648RSPDUQ2XOoP5L9NeuZt5V1fT
1VvWAnKCBs5zttISNVp7tPu8DTgLRh4S/7M+gEUQB1Ogq7fyHJvFsCqGEY32me
61N67aHkqMLa06ZI9JIK/3SfIPpzw4tk+sQCqEazpwVtkT7qF2+U81Vc0XXKjLl
axesopsULCG886TEApoyTj9jk1i6ZjzmgRZLUsS50g+GdRTPaFPxMGxjctU7U6cDIw
SPA61uxJp9XCa/Gs6YLFas9VBhce/80C714syjLDcCgYEAgAG7yuM/CSY3MrArw8f
f4W9qkgIgtwfn2p2jqbttjEK8GOKXvcle4Q9QaJoiY6HPZM8hp06kUjSCyZxGcKF
s33YyZ+c0rzZtquX3b1QA4tNFlU50P0F0En28KhXSImbhXxgb+L8mJNUp6y1uEw+S
cuQxAliey0Gjes63hpl010CgEAYuEicILQpTCoMyAYWgC93n5u6i+Ic089sylG
ewihrakhiLWYTs3xhyGwKbA4101wEOwYV77DPDOs3tCIezhN8WKn7kTAFj1HO
YZeZMsFU9l1utPwQuMwq71QkqOkeIRrCOC8D8sRdW815Q45uKjnmmkSpbHjd1FNKGe
4qfIK40wCqBYoAVt3sfLmMkGt7s3hKz4hZM5FbLP/iMGWZKMAzDW+XxswvB6lbrCWY
8c3hbohuJbpyZa4qR2G+qtRmodLca+tQFMrOEH/g1NCY+WoHSRINhmcS8w
dsszgFwFlnxRLb1DF1HZwSqb2tLYBP1HJf1TP060UT8aD4Qpw==
-----END RSA PRIVATE KEY-----
```
Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEVBQUAMHAxCzAJBgNV
BAITAlVTMRMwEQYDVQQIDA0DQwDQYJKoZIhvcNAQELBQADggIBAHCn7wzUQ/TJ
JG7Q4W0S3sFzvQ4l0kvoHwqLUGQrGj9Z7w3JWZ/nRl9QZB/FQRzahs/9rHfz
uA/PkK9GcZdHq/PdB6c0vDn0m05rYc7YVtNf2R3T6g/eZ06G6l9PzG
-----END CERTIFICATE-----

Private key for domain certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAoRW5jJmVqp+qSz273JZBGn1Z8j5WxCAakkfe8B9qy0SUJ3GUFXv5HYPJpD8adcCTSUAlKIpFy88erk4vHVwCMANmxa2xzcG/ArnfXM37sXQLL8whIywSTY136nLw3deGNYM15SWMCHqgM7h/FL8EXLz0jMAPG885+kTjQQAEXlqPt77GQ2QSt2QzJsBxJkJncgc/fA/MGYzMKw7D9+Lx/72PUajQO7PTUX2p8P6nP0rWYfGBPUxroZGWS0bg1rBKK1fVbfVBsBF+P3/+FHtmB4HSA7ossZdw2d2Xzs10dhckxk2Kc3gk/xy11hJBNd1u3W1fSa2rt6U+ySKKcO5SQIDAQABaoIABAB19qI2A0ed2LxJYCja/ON4EBbrdHluumvOnecIc/J3XjTD2Nnt8T0gdJUJPdhjjwZ2Qzz7kYdzDN4j6Akesz30sT2MTFeb/WiCT6cAH1VrrK33cK6zYY217aPj1H8IUaUr1T73Unt/DMp6gMcFbo+XQZ18evFrc8zubc+BK7KsN4N6b/zMhw+PXEiy2EGDN1F04TMhxPD4w8IMU8oL1L8A6GKImxK3gMuIiS6Ruuau2hP6kjkhhKxAyzU1s8BCMoLDoJjyyH19P6Isrv0NFve0gMO2Pdz/94ynFpDmBXTqBbAbT09eicyulK1L0g/ERmj6jiImsG3YRWEEDGzH3xUOUCgVEA0FDEUek2LHyU1Txw1zhDTldyuItiYzQ/Mexa2q2eA96zhJB16aX+55PQIxEefhTFn4e4cKjXQSD7aixy7jpkKGowFRL1BpwbLduhlniYXsa8Kv0OpJM4DTAq0e4FQz5243KH755U57tjCEdOni4j4DA3GnPjQKx973pdyVCqYAXxFUx/zMXgT7pHxW/QHZD7xXEs4Fp1xjzL5BaHoJnM7WbnkVwUvcMaEE/1r9gyGI4XRNjX6KBZ9UVgh/B0/ACyma3DImTa0+Uie9kn7TjT5pzwIAFh+FyQ4tULWr5grczrvPjG9GtXmthu1bILSumVEmwC+P6kailr4xplez18CgYEAJf51sk2c1mqnpxJXjmLjbdasA+w6ycD9mluqaXG08uswmgQ70KrsphemOgQFVjisjPnU27LW1/AKcDvzkeDdU4F5xT4J4D3zBq713fxQRXVb+3ZYvfN0vcWsc1VnjeRgP7zsEs8R8ufhtFlrRPOwWn6qmyQVjnTkCgY/3zlinKBkq90eZU3Iq4TXL5pLeMo0oFOcjcjkjVjvnTRcXTM5npgPSEaiL6QOQ+s6OYyGlnyV0SwLPW/VVb8FHHj3lzWC66vcKucDz5JnFwGmL1bzywTwa9choIj0JLW51j1gHc9Uu0h0V9bSjyr69e6+H4v4NvUOholcQxK6GbqCKLEHCvQQfWQX8fkgw+YKnmmA6o25ZI31HUQW/YfscoCOVnWo+vwovucuTTtcbW1kXtReEjPufE6p5NIZkmsNUIGrCpx/3uq27JFMCpJzJN9b1FM4uc1cKf91ehiYIFVfNvxq+dVnxBGcXknhYK1Mnt9b3K6mDqerQjK17KryaqJZ2qQ==
-----END RSA PRIVATE KEY-----
Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pUAMA0GCSqGSIb3DQEBBQUHAMHAxCzAJBgNV
BAYTA1VTMRMwEQYDVQQIDAPyWxpZm9ybmlhMRQwDwYDVQQHDATYW4gSm9zZTEOMAwGA1UE
CgwFc2lwOXMzKTA0MBwGCSqGSIb3DQEJARYMIEgwFQgQ18GA1UEA8MDQF8wLQYDVQQE
MDExMDAETMCIGA1UdDwQEBzAFAAMBAAGCAwggEBA0wHQYDVR0OBBYEFC1TKpLjuKa/dP
umVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8
GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0O
BBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/
dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6E
MB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR
0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuK
a/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR
6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYD
VR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLju
Ka/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR
6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYD
VR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLju
Ka/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4U
6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYD
VR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLju
Ka/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR
6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYD
VR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLju
Ka/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4U
6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYD
VR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLju
Ka/dPumVbeFXEW4UR6EMB8GA1UdIwQQAYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4U
-----END CERTIFICATE-----
Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAqhbHyDUpucZfZqS5etuh3altHR+mrHzXs7+0hYd12x4AIuWq
o88pyK1rAAVtna5/t13k6ReGSBEkbDcG68+1Q1LVnMOPGwpJ32a8sxQ+t30kVrN6H8
DAGNzTA0HP/q8RZeuvb8XDDDNoKgrD7xXbghuYyds2qaW3Lso5edi3qasbLocn17
ri9k9DbWp7TN7i1x8spyk5u5X9MXvm61ucfbnMTQL+jVv7HV93DIOwFQ3FS4Dvwsj
Y12SU5C4foxa/Hhy3B1TNQ1lJd77MBGJH09LaRv5bC7QRE17uEicsK/7nr82814
e5zXw4a0pJAgLqTvgsIPsFvmeW2ctWAns7G3bwIADQAAbOIAH1jppv+B5YVTIL59
+Ucr4fJyKVGLIoQF/CygafJztZTVa6v/a/n8Rkg8XyrJ9aXvZVBIQ1ubdM4Z9iN
8faVSKLAsj3thtkfSojTMzU77x+iDcG6LxSzekAGqAIJ7sRL+iEzl/Fm1WlqEYh1
GIWIlgHH01n300eCy72dwmAV+2aHzn8eBggkXwm0fb1RC9pVh0FC0+jy11HasjL
0OBkH511bmz4PUuUY72j2665gPm710nr25igeF842JkJaBqAV8Ar0lQ1X7YLee
6QyLvoodeb0rHz8IezahWAdmIPGCIUCfM7RmyInOatGAo0VEU3uYnkUQQVoi+JTX
46CCmBeCgYEA41Cd/IVz9pDw1j/0Ma9J4zeF7Pgn5MdxNjmsSxvHSMINwul1u
BCYozts77vWbiUxiX02Qse9maGA2ss3+VnxB0eu6EBQ/fK1CcQQQH52nXdrVlsqbnK
5B5e1FKcgZKpFNVwr0BC6csDndTch9P5IKsxWkesLC3VzSUXMsmCqGyEAWNYV
+SaC1QGTL8Z8Z8zkyE2nhqRUFKnc/twQJop5g9E4w3Lqj13 kýCYUqSdYExlxQDE3
6C0m17Jc27jggD7qrg1JxMznRXLMeg77bf7fEP/STVOH5uagEB7ktF18xJjKt
yoCK1L11l1QrToSa4uetHLRXKCDSEpRlsVw7wRdkCyeAeKDBXyA/nykYDUqPdi57
1pbFkDD9G5x+YVPTUoX6wpugapF3jENAHzVq00dTRDTrYm8T8dp22WiS3bA7WS
hFCCTfVweczM+ID29GKnvoQ761m6C727j3S6fEXBUHePae072XZ8DCX1dsmEJeN
+M2KhxcGx19tIehJ31fouykGsY9B4us1PwaTeVX13ORDuyhUQ0xOoNMMA491Euh8
FpCpiDZtLmzyZVWjPeLXFqWqUgLmHMJZJeNeRnFqPcqRlr56zqXrZ5j/wBEnL12Bm
cTXLp6vnPfHJq+xo4eQ5HgZKgYbV1hHs51CyD+xoP4sWExpmN+Gdn2FCXywsAF
UCJ4QKBgAKSrm85YqGhd8RAAmd9JZLGOlPmRXN98f3FEnkX7JZEZasn18vD
65z04h5cohJ3XkxJ6k3lC3WopzVrs1ha32MEoJPCgwBsa8LzrrL3YQin6yf
+baMfTDMhiopORB360DY41kcwKxzQ0n3XATlrl7NRV5wHr2ejkY
-----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=sipt,
OU=Sipt Test Certificate Authority
Validity
Not After : Jan 14 20:21:13 2011 GMT
Subject: C=US, ST=California, L=San Jose, O=sipt,
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):
dd: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:dc:9a:45:0f:28:8d:dd:fa:15:56:d5:35:17:
f1:5c:f5:38:4f:36:54:8a:b6:7b:6f:ff:85:ff:8d:
80:bf:dd:44:ff:ff:39:0a:2b:ee:4d:3f:5e:7a:aa:
34:e5

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

Signature:
c2:3e:00:ce:5f:b4:c8:da:ab:b5:2f:cc:2:89:60: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:e9:9a:70:be:f7:4f:08:6b:4a:db:44:
9f:35:6e:ff:6a:8a:80:74:6c:f8:a1:0f:3b:bb:2b:
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:

Jennings, et al.  Informational  [Page 60]
X.509v3 Key Usage:
- Digital Signature, Non Repudiation, Key Encipherment
- E-mail Protection, 1.3.6.1.5.5.7.3.20

Signature Algorithm: sha1WithRSAEncryption

Certificate for CA for example.net in PEM format:

-----BEGIN CERTIFICATE-----
MIIDzzCCAregAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBCwUAHMAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQDAhoGCMIEA1UEAxMI
U2FsaWZvcm5pYwYDVQQDDA9UZXN0IENBIGZvciBleGFtcGxlLm5ldDEUMBIGA1UEAxM
LZXhhbXhBdBEBMBQGCCsGAQUFBwEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABG
CCsGAQUFBwEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAg
EBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEB
BjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBj
ABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjAB
AgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAg
EBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEB
BjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBj
ABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjABAgEBBjAB
Tm5ldElEIPoQGwcoAEATCvA19jVysh5UxUu1FqA5vXwA8GjA9kwY2uMEr该项目的TSGcN
-----END CERTIFICATE-----
Private key for CA for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAkCAQEA1EZlUfiEHLWTR6UVFAds3Cp3kxFedRTsIfS9F1DdT9+KuQm
laMz5F43HeWNNwxcwYc7k/0iEnF008PeluHwHi88hSAXF0r12Q+BzhC18XWGSa8
fQv43175D9yaRq8ojo3d6FvbVRNcogNL8H9aVn1U0LEc4u639D19o0Mzq101JvNpI
2NyGQrUmzB1SVAhS5799dpV3jcbym/M/AHy8j8W1RdW3WxXPF19D2UirZ7b/h+V+Jy
3617VEVdkg+qsihqdGtvMnIEKhig20VVRZD8DFKkSR2GelrBpGmljbt53zkQNoVwE
K4cV2cIRAI7CC1PCAv91E5Y5CivjTdp156o05QIDAQABoIA6Dp7/7/PIH79vecn3
z7hGNE50aG8HuroPrSh3yJG4a+067XbzaRW2I3ZxuaiEHiexoY7dua97xu4dbJC
f2J1jr4UI4Is4aV7Nd5W09VNg308NkWwLEnV288Eo2Tgqc8wXz/BleL9nCJWcH4Y
Jw1rKkXmkTdQpVBCwcPlJ9UzduXq2fBbrsL6+OZ+F3kbvUwYAVhhUuBS9sf4Xib
5GAA2CDLpm433giO5y9KigpcLvhbAhMiPTXJ6i65m9xGgCjxhjP/3dr0H0CnczRD
yWOCfaNRJug9kEvn+n3uGaVFonU79cqb1FXG07ea7G+mfp3Cf744kFEXz04k
8WLW6gEgYEA91K9MhWt6B1+xPJB42aA5QvrF7nLt8e7/3aTncyMI013xUyPdpj
TNEfqaRbptmwd2HvTtXj1Q54fE+pE+q8s00Rh2FVoWi91zI4C8Wnm/6J5p+QiXy
tc2DFP22bmsSWuAqyOhufIMhzoek1BbUH5q5YrcaA5DmmQtaxciz+IECqYEAJ07
6DamIgy0eJ02GKHU/Hy8RvQ2gauZCtmgmLQRWzeOmx9hQRela71Q55F6Y3HQRcTD
RDDjUa9Y8BJ0WTkasbRgxjmHlq1f4pUdT6ycfWgsbCNFTosgPH+/OZPEH4DKL0
rb1dUzHPu2dco2Q72KtSPMK+ikey21C29cm2nKMUCyEAsGoX4fJ/HpDMzrcKf4qTG
CoBbojX2+wPhVTFV/0LtBwTCG3y3CrpG5yWo41nRWPFEQmWuW9cn+EN27JQXLQ+4
7vPvly6r/vOSAMA9sCswW22tB3Fw4v0qFR3W3?3AtUECgFtNkb+jqFX/QFaH02c
6Kxw5M5vqyqTjX7FVycp51ECgYA4Tq1WpHqcpq99Qv4sJUnuM4v+dBj6dq9Qd6qNF
HEUgNc2B0C5Wx7D4+rxM7qWNC2t3S7N9mKLRRwBeg2r2xv0FUj7y7i0cmiuE
BNWYfoqiS7fHvA3hO0Nw/Ezq6J07v0XfGlUtb9b4+VoazHYYE83Gg8H7pjcXw7d
qD7L/QKBBgCedLkX51d/EqW8KwK5q/b1G/T0zu3MCD1zCjfa2BHMawSRALD+unMAMDEA1PHOFs7FmCfsP8NY7+W15/k9WupwQFSTY8dSRVdpFp1Fr8t25yX2
mdBrU3jVs1AQPEEKPbOlXpLoEGLvoTHFWSzgmcCIKXXq0L+0+w==
-----END RSA PRIVATE KEY-----
Robert’s certificate:

-----BEGIN CERTIFICATE-----
MIIEJjCCAw6gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMH0xCzAJAJBgvNV
BAYTAlVTMRMwEQYDVQQIEmdpYWxpz2m9ybnhMREwDwYDVQQHEwhTYW4gSm9zZTEO
MAwGA1UEChMFMc21waxQIXDAeBgNVBAsTF1Rlc3QgQ0Eg2m9y1GV4YW1wbGUubmV0
MRQ0dGhveDVQQDEweFGtcGxlM1ddAgFw0xMTAyMDcyMTQwMDEwMDEx
NDIwMjExM1owVjELMAkGA1UEBhMCMVVMZExARBgNVBAsTCkNhbGlmb3JuaWEExETAP
BgNVBAcTCFNhbikBKb3NlMQ4wDAYDVQQKEwVzaXZpdEPMA0GA1UEAxMGcm9iZXJ0
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA09wUaWtxCSwLD52VCQFK
IGbvn5wwbJ56xQW2hnnMQU2xz/hTW6UNduyXuhYQn+1IXtfttS5+P0J8OFAc+1MTk
7zXRY5EgaBj0jju0DwM+oADWwybnV4hkhqN6LSFESNaBuT0o3Nbj0bPySyYPP9SZ
Y+R+FAqyccxfO0E262pwwdPCGtK2Q0C6tLQZiy1LEwefE0ncOpA7coFSJdzaKt
qwf2/2n07Lp/S7b2KIwhNh+Jm0SQ05Xe+iFhJdkw3NDq82zW29ShgASlc9B56FRk0
gRzPgcZcd/mBZ9xQCeKo152fNW7/pgiAdGz4oQrzuyu2UYwhvA2ydCVqtMCLM75
IwIDAQABo4HNM1HMMEGA1UEQzRMRDoGhveDQPGmDpyp2J1cnRAZxhbxBfS2S5uXSG
FWltOnJyvrEBe1eGFtcGxlM1ldIYXchJLczpyb2J1cnRAZxhbxBfS2S5uXQw
CQYDVR0TBAIWAQADAgNvhQ4EFgQUpkK99Yg1rv+51nMe8CT8LOhKrc4wHwYDVROj
BBgwgAUCnDPZ4jptj8b0CPohqvy41 iv6cMwCDYVR0PBDAQAgKgMB0AgALiuOQW
MBQGCCsGAQUFBwMEbgrBgEFBQcDFDANBgkqhkiG9w0BAQUFBAoCAQIEAJZnqGh6W
bU6xnFdpd+o6pG3Itu51joeF/cTLrLKqN3JpdtqCqLaZa40/JpGdxbq4j0dIdZl
leC4A0F525m/4ezzzMRRsdovrjyjh7cZbhOere5jzsb9PjowUX/ZWmtc1jrUuM61
izF0cLPMXASQ522QdVX7wdjc28899gOSNL365K6Kex5v0E5u9/CmYTuemO4aAQ3
rzZMZX9WWhI1YuGnW/2981Q0Ae5ZLnx7ubVQ7yfwXBlPbGAKYesloy7BGLG7
7w0qYGomaQ1cWjzJtC1l8d+AM5DyYFNI+3plyV8fo+h1QkL1rdtGcYPPGAsis4
28xUy7kNhSVA==
-----END CERTIFICATE-----
Robert's private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA09wUaWtxCSwLD52VCMFKIGbn5wwBjAs56xQW2hmMQU2xz/hT
W6UNduyXuhYQn+1XfettS5+S0J80Fac+iMTk7zXRY5EgaBj0ju0DwM+oADWybn
4V4hkhq6LFSFNSbVtC3Nbj0bPySyYPP9SYZ+R+FAqycxxfoOEZE26pzpwwdPCgT
2QOC6LrQziyYIEWeNgEOncOpA7ccoF5jzaKtqwf2/2n07lp/S72bKwNhN+Jm05Q0
5x+eiFhJQdvw3NDq8Zw299hgA5c1g956Frk0gRzPgG2cd/Mb29xQCeK0152fNWy7
piqAdGz4o2qZruyu2UYhvwAzyWdCvQfMCVM751wIDAQABoaIABAv+q3GMUYPrAhbj
lth+EKw86MfCUBn2689rjbeFC38j0QAa/CgkAGPfK7zBFwnY8R8YTXJhEAnVbW+zB
4PpPhwonoUjfqFP84PavffYVN5nsrBByWtdO0a4mDnBf7vec99u17KX5j2HN
r8NPRTt6a800xdAf9YGv6WetC0hK8dHhQyM YM/ Vu2Kp0f01hTpFlmxS7We+duQ
mva15UC8+EL0799uhokch404E0036Ce4luCnqQfOUAkCXMCKY7215gquipe620IXLE
Cqee2ZPEn8egWhSNG1981CF15aAeB0tApMcMwrfcbpnQMHoqyQhm2XeqfogQGIN
UAo15NECqYE9TrfG9wu1vfi+kxtX61Mjw07yNg443Ntb/9+8xKcOiz6LopBO
VHSVgHJbijcSBuua77Kx61HAv7A0v0s2FRHAb3M7wOYVgK5T2+12o4Fh6EMU42G
ISAcsS4vfChHyt10C91B1YXxxurp0yy1rkEaaSLHAn6rAEwCxcQyEA3osod
geCah7Qenu5P8U5j9yFbArBqvdXQKmn02trrkLkyVtgven7ES31EoGeoW5hr3nr5Isk
IpwFgBq1vEvEGUv3d0Jc5sZTET0ewWBLec/Ct2zfnhBcCNX8jwX5mtCt7ZMhuxV
Vj1WpDn+K7+G8KIK0+Kp5QdOCxXpThrLHgBPCBurCgAvYgCulFL8B3QVfQSpkL
TZeap5kbyd7ZIFz2jPzj4P+tonr87TafiliPogr5gI1WLSL8BNTeZeYrQsrs
iuwW39ezKXVmsuasAZrgNob1Jaw6lJx0j4046m413x7eUe金陵/J9C59SOLQswj1
2+fntWpmP8drfe/5Sfh+LQKbgQcyDaf2Ke/cHCmiuxHuvUrshs4kccTGofe75Rdi
hqNdvyPZNHV9uKsrtnv2Y5MJPJeGskF+Qtvkp31lyqShQv43HTI6nhb/A5p5bb
//muezQxqj9k5UMkELOJdNnBcFk/fH626W7qF5Ec/hFVvG03ju07PBcAYy
CuSFQKbgBw2k55Q5Dv4u1wVe4wGEi17ia+14lzmg8wpj1DJUNoxCdv1dGzaA9q59
wP0r+vjhK65Vrl1m0tqcC7207pC3JjBEtAchKhj2Ikt+ZAE1jQmF+F62Rcv6Sbozq0
5dFCA2wZe122qomq3j38+OyILUs/uzFkGj1oJrJr+OtpPKsrcr+/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 | tar xfz -" would recover the CMS messages and allow them to be used as test vectors.

```
-- BEGIN MESSAGE ARCHIVE --
H4sIAIpaUE0CA+ybeUTx7hHCSiChi1poqS1qvFECu5tsDhADEATQhCQeEXTZ
JBT1yUGSIEREREU8i2L2QzvYERHVCkIUBW1vusXCJeeIV3LfpCRAuUF58f
xjXH/JPdm3f7fYz8+nfr88J6L6EKSVCCyyTfKMcy+Hypk/0LAAEBAggb8Eki
wp98NhsIQAcxIAhDBACGIRDCAICBQCTqYAGdv6HKEF1QtsVrkhKSD9kZVv
jdvF+1+HcZy10r+Bgd5soX1im0u0fHSITRMnDujykJyRtqQw4b4TApjNoy
Vlg4/37mxBwtGApU21hNyBfymEAApE42CmKTI3U1JKV0Bo5YHJ9MmgkaHUI
CxsAgSvAc3kwgDqGqzBu2/XhYvumUNCJAIQmgQ4adUe08Z04RMFXOMNJ2hqm
z8k7qvu+uzn28FbiJL+IC8QXAKAH8h3aeToLmokIkXWISQaEHImPcYgYjHo
1+qMYui4gdspw/ky9mM33V2mOAWTDbCpQ6fEsSpussUpOybZIraj9rZg
```
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