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):
        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: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
94:8d:97:4a:00:94:00:bd:25:b8:17:2c:52:53:5d:cc:5c:48:
91:06:c8:0c

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

26:1= 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
63:1= 10 prim: UTF8STRING
53 61 6c 69 66 6f 72 6e-69 61  
California
57:1= 17 cons: SET
75:1= 17 cons: SEQUENCE
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 66 69 32 61 6c 69 66 6f 72 6e-69 61  
Sipit Test Certificate
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
53 61 6e 20 4a 6f 73 65-  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
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
267: l=  32 prim:   UTF8STRING
53 69 70 69 74 20 54 65-73 74 72 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 82 01 0a 02 81 01-01 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 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....
92 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 ??????Fwl...>.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 lc 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 .kyy....Y......1
70 37 a3 3f a5 7d 0e 95-46 f6 f6 58 67 83 75 42 4p7.].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 97 50 cb cc 45 ......s..P..E
34 23 49 41 85 22 24 ad-29 c3 02 03 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-6a 65 98 12 91 04 f3 63 c7 ..E_.+..c..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
30 16 80 14 95 45 7e 5f-2b ea 6a 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:

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 (EKT) 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_CBC_SHA
  TLS_ECDH_RSA_WITH_DES_CBC_SHA
  TLS_RSA_WITH_DES_CBC_SHA
  TLS_RSA_EXPORT_WITH_DES_CBC_SHA
  TLS_DHE_DSS_WITH_DES_CBC_SHA
  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  TLS_DHE_RSA_WITH_DES_CBC_SHA
  TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  TLS_DHE_DIFFIE_HELLMAN_EXPORT_40_CBC
  TLS_DHE_RSA_WITH_RC4_128_SHA
  TLS_ECDHE_RSA_WITH_RC4_128_SHA
  TLS_RSA_WITH_RC4_128_SHA
  TLS_RSA_EXPORT_WITH_RC4_128_SHA
  TLS_DHE_RSA_EXPORT_WITH_RC4_128_SHA
  TLS_DHE_DSS_EXPORT_WITH_RC4_128_SHA
  TLS_RSA_EXPORT_WITH_RC4_128_MD5
  TLS_ECDHE_RSA_WITH_RC4_128_SHA
  TLS_RSA_WITH_RC4_128_SHA
  TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
  TLS_RSA_EXPORT_WITH_DES_CBC_SHA
  TLS_DHE_RSA_WITH_DES_CBC_SHA
  TLS_DHE_DSS_WITH_DES_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
<allOneLine>
Via: SIP/2.0/TLS 192.0.2.2:15001;
    branch=z9hG4bK-d8754z-c785a077a9a8451b-1----d8754z--;
    rport=50738
</allOneLine>
To: <sips:kumiko@example.net:5061>;tag=0d075510
From: <sips:fluffy@example.com:15001>;tag=1a93430b
Call-ID: OTZmMDE2OWNlYTVjNDkzYzBhMWRlMDU4NDExZmU4ZTQ.
CSeq: 4308 MESSAGE
Content-Type: text/plain
Content-Length: 0
Hello!
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: N2NiZjI0NjRjNDQ0MTY1NDRjNWNmMGU1MDRrYmI.
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--
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’
```

SET {
  SEQUENCE {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    UTF8String 'San Jose'
  }
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    UTF8String 'sipit'
  }
}

SET {
  SEQUENCE {
    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 sha1 (1 3 14 3 2 26)
  NULL
}

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 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 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 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
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 {
  OBJECT IDENTIFIER localityName (2 5 4 7)
  UTF8String 'San Jose'
}

SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    UTF8String 'sipit'
  }
}

SET {
  SEQUENCE {
    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 sha1 (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 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 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 88 E8 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 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: MDYyMDhhODA3NWE2ZjEyYzAwOTZlMjExNWI2ZWQwZGM.
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] {
```
SEQUENCE {
  INTEGER 0
}
SET {
  SEQUENCE {
    INTEGER 0
    SEQUENCE {
      SEQUENCE {
        SET {
          SEQUENCE {
            OBJECT IDENTIFIER countryName (2 5 4 6)
            PrintableString 'US'
          }
          SET {
            SEQUENCE {
              OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
              UTF8String 'California'
            }
            SET {
              SEQUENCE {
                OBJECT IDENTIFIER localityName (2 5 4 7)
                UTF8String 'San Jose'
              }
              SET {
                SEQUENCE {
                  OBJECT IDENTIFIER organizationName (2 5 4 10)
                  UTF8String 'sipit'
                }
                SET {
                  SEQUENCE {
                    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
                    UTF8String 'Sipit Test Certificate Authority'
                  }
                  SET {
                    SEQUENCE {
                      OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
                      rsaEncryption (1 2 840 113549 1 1 1)
                      NULL
                    }
                  }
                }
              }
            }
          }
          INTEGER 00 96 A3 84 17 4E EF 8A 4E
        }
      }
    }
  }
}
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: MjYwMzdjYTY3YWRkYzgzMjU0MG14Mzc2Njk1YzJlNzE.
CSeq: 5449 MESSAGE
</allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime, application/sdp, multipart/alternative
</allOneLine>
Content-Type: multipart/signed;boundary=e8df61ce5d1e864;
micalg=sha1;protocol="application/pkcs7-signature"
</allOneLine>
Content-Length: 1455
--e8df61ce5d1e864
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
    name=smime.p7m
</allOneLine>
Content-Disposition: attachment;handling=required;
    filename=smime.p7
</allOneLine>
Content-Transfer-Encoding: binary

***************
* BINARY BLOB 3 *
***************
--e8df61ce5d1e864
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)
48  2:             PrintableString 'US'
50  2:             }
54  19:         SET {
56  17:           SEQUENCE {
58  3:             OBJECT IDENTIFIER
61  1:             stateOrProvinceName (2 5 4 8)
63  10:             UTF8String 'California'
67  7:             }
75  17:         SET {
77  15:           SEQUENCE {
79  3:             OBJECT IDENTIFIER localityName (2 5 4 7)
83  8:             UTF8String 'San Jose'
87  5:             }
94  14:         SET {
96  12:           SEQUENCE {
98  3:             OBJECT IDENTIFIER
101  1:             organizationName (2 5 4 10)
103  5:             UTF8String 'sipit'
107  3:             }
110  41:         SET {
112  39:           SEQUENCE {
114  3:             OBJECT IDENTIFIER
117  1:             organizationalUnitName (2 5 4 11)
119  32:             UTF8String 'Sipit Test Certificate Authority'
123  3:             }
```

: )
: }
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 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 0B 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
: }
497 256: OCTET STRING
: 88 9B 13 75 A7 66 14 C3 CF CD C6 FF D2 91 5D A0
: }
507 52: OCTET STRING
: 9F 11 0B 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 0B 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
: }

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 {
   :   }
```

Jennings, et al. Informational [Page 26]
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


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
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
[ 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-----
MIIFDjBABgkqhkiG9w0BBQowWzAbBgkgqhkiG9w0BBQQwDgQIwtc771D1NUCAggA
MBQQCCgGSIb3DQMBHABArd3Z1i2TawSCBMgXoO/H/dTPlHwnqfW7Uh1dr7767zB
1sXxlenMA61ymALF/4EIteqOE2/aEbr8W3tVpjNpexw9r5TBsbA1I9/FMMe+USc1r
5pIdDLx7yZhXvC0UW1xbGWlEcLMgXOvzkwW/oGq49Yq1ce1Gb1LSV2L7Wi93TUQ
Q81510x7yjx7c7kaHOTyana0nxUX1qUL25xTbHWNuf1nPEzyEUIo6Ir6TFU071dShnXb
RogQqV+5owsn7zwfyGz3QocM9WaZwKFOEOqByFgaaZ9mL+cn1rZ+1/IDt7cSB1R
3ucN2mGdEVUvzSACZ2L9fUO1707BQM56enDRsqZji4WgqDHxa4gkJkqPJeBnLVA
jxCM5L/JyikM25iKHMd8LWOCkO/Rk+799913Qv1Ymn7yCincorqdi1rAdmq1Z8Tj
QPqXio1lx6+6xyiDCV7Mwkyo3X1k9y/Tf2cZ/\wUwof/1fMaag8HfSPN14Rkqsz
ufL41K5scPZPluQdo0Q5SGPC0JgsckPcfi76zr162KLPYrwc65HT9PdevQvC6O
VgqlxkEGJE1v1llm62/G0tQKIA6bh8pszvwmHjGo9s+f+p7KJYyV5EHNEH5Rm+T
8M2owk67033sV61CIDOArdRL8oi7hmmM+r1x9VVpppsDrzjqQqVYGBbEJEW8eQp
t7kAjuN48TDD1mS8E6datFv/650AjzAgCbjkuPJoWU5dF1Y+i1o9vcunocjh+1
KVTXsM34wOsBMPJBfQ+Aww5b5sIEkVIIi1OLYavf17/FbPz20gcdP5wM53y1cblKL2
ThJV7miW0vY770aQypnBaU0Ak90zBvEvPNahrDIINucbEKFrhN2pfO0s7k4lurjik
uknkRm3g0cDdtyM2X810Yn0x6HnpC+hoSoSRoR/kd96sAsapy6g9S9hj1lnBnbc
dlq/f0/1o9MDujT/hujZ7FqdznM3K4A6vf0kmmVM+GJyBce+cjkX6W8B01f91ycB
0wpF+4f2wL52F5uFcyvU5WFXvbr1+1MkV6sNdKcXXHzzAE61nyU19bwVcAlsI1
WNgEFhntb1wZbeW+3gH80v12XVCgEmakaHjakSaHjKXCGsSx1x2FStznpVfbPbnw
0yrdm9xwegE319aRuvR6gdf051Lz7XjvvrV1CemPUT6YRBamFNCBo7C7cjgiE
kgkMQFowK_MYEOy2ddZMnGzsSPKk10n153RqPyd/8FT5dPuq073SjxyKThAvb1+kVL
1rf26b7p7/UKwLBC3tBLG6u/Ez84euWN+U2JXIAcDoCCWeRqkgf4j368b228dd
A27X42J+q+yFshFNOA7vhSiH3Am3gBzQhEEqRsRqgk8gqt1Rghq1823GEeexxUf
8s100oju008HqAhKTrtwjV5+0C6Q6R9W9Mnwz7mszHoK7QT8kzL7KUXwb6DvWbc6W
/UTugxVXghA8hjmsv9e9ftDKL6v9z1p4vRVRdDzm4TYyubYh5ui9FbJFJLNjJno
TcnnusHO8Cqgs64khLRmM46oi+JSEPv7o7zHcFwNOvNTWN980EKCBtED2tnQn9VC
0skxy9/R/znunaL3G3l2RuhWpyyyvdxN1q3ie4tcrMLXIE14U2NOsPCKZY//NEn
BEc=
-----END ENCRYPTED PRIVATE KEY-----
EOF
cat > demoCA/cacert.pem <<EOF
-----BEGIN CERTIFICATE-----
MIIDtTCCAp2gAwIBAgIJAJajhBdO74pMMA0GCSqGSIb3DQEJBQQAMH4xCzAJBgNV
BAYTAlVTMRMwEQYDVQQIcAgDMVQHMRMwEQYDVQQIDAtc2VydmljZSIeIzAPMB8GA1Ud
EyBjcm9rZTCCASIwDQYJKoZIhvcNAQEFBQADggEBALJ15CnRk6kVHEjJwPZ1wofCFcN
h27gyjg1b9R7YQ+6L/uVzH2k8K5YUBc4mANl+Br4CpERLHOGiO16Iyv0oLwOG8QsA
57USh0QV7oZ0vIyj5kJG95vAjvRb/4QHaajxwi4x/jwS7/AOYS+F+ZsM2Lan14OOGm
01 DOCUMENT 1527/K3DLm6/h2ZAE1XVnJ4G63Z8Yh0IsoYf6V3b6V87K7jy3BekmBb
-----END CERTIFICATE-----
EOF

# uncomment the following lines to generate your own key pair

# openssl req -newkey rsa:2048 -passin pass:password \
#    -passout pass:password -set_serial 0x96a384174eef8a4c \
#    -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
# hexdump -n 4 -e '4/1 "%04u"' /dev/random > demoCA/serial
echo 96a384174ef8a4d > 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
" if (/\w+\..*/)}'  
USER='echo $1 | perl -ne '{print "$1" if (/\w+\@\w+\..*/)}'  
ADDR=$1
echo "making cert for $DOMAIN $ADDR"
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="URI:sip:$(ADDR),URI:im:$(ADDR),URI:pres:$(ADDR)" ;;
  *)    ALTNAME="DNS:$(DOMAIN),URI:sip:$(ADDR)" ;;
esac

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

echo export ALTNAME

echo openssl genrsa -out $(ADDR)_key.pem 2048
openssl req -new -config openssl.cnf reqexts $(ExtPrefix)_req \
    -sh3l -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 sh3l -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 sh3l -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; \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].
Fluffy’s user certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pNMA0GCSqGSIb3DQEeBBQUAMHAtCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpz29ybihMREwDVyQHDADAhTWY4gSm9zZTEOM
AwGAIUUECGwF21waXQxKTA6BQNVAsMIFNpcG10IFRlc3QqQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAxDTExMDIwNzE5MzIxNloYDzIxMTEwMTE0MTkzNjE3WjBWMQsw
CQYDVQQGEwJUZXETMBEGA1UECBMBMQ2FsaWZucmVjdEp5YTERMA8GAIUEBxtMIU2Fi
Epv
c2UvdjAMBgNVBAoTBXNpcG10MQswDQQDVRQDEwZmBHvM2nkOggEIMA0GCSqGSIb3
DQEBQQUAIAIBdAwgEKAoIBAQCjLFKm6bzk7NOe+5kC7LE2orFTHU3DorauUL1f
VqH3jH6k6fBoMSiPZjJWgcMil6dt/acIKgGl2G9X37BFOwYKbQ0TjKJu4N2tzn
uXjWrKwEeDkYwLnXnarctszzz65e174tZQIAx/6nag83p+fjH7CN0H1vbBRCxgo
4Y/9Vkl9zxbcqcqVhCwrKyuxR7FNuPSsAgP41GwYKROICOTzzF0rDrSiC6ChhBQu
7ivjL8EanoaamGqiTFe5wMo1YNVbAv+NrHPAHcyy0xjGzGXLrj6LkiQBQ/rft
EgGXXHUF4ahWL+yeWc0RGNNNckHM9wxjds+jpRb8KZn493PtPLVAgMAAcgjogwcow
UQYDVR0RBWcwoSYWc2lwomZsdWZmeUBLegFtcGxljLmNvbYXVaW06x1ZmZ5QG4
YWlwbGuluY29tthdwcvmZodWZmeUBLegFtcGxljLmNvbTAjBqNVHREIAjAMBG
A1UdDgQWBBBSFlwn401U3Jlrc3uORcuQiz5iHUJafBgnVHSMEGAWgB5VRX5fK+l
mBKRBNjx2iaWmB3jJzALBpNVHQ8EBAMCBeAwHQYDVOR1BBYYwFAYIKwYBBQUaWQ
CCsGAGUFbwMUMA0GCSqGSIb3DQEeBBQUAIAIBAQCqY/Yigu7f9Pv+XNj57u0oby
LKrjIluacV7YK1Pd2dp8ujvwmfhMbMyKe0+RK43W9xT7eBeBg0zRQvL5nIs3hDv0
mmjuiUFLyC21i8wDoENXsO0hF9Txl1RnVvETzy/8i4PFOcBglmDzLCN8Mfa
TrHzcFPTbDtrHRm2H2rsb/h/EhMjHgzb9bX/XasVDUmlkQA0kNvYBxGQqQ6E5iq
nrBi0zwbLcvpmSUjyFAPwGZvc3RqjjuRzgeyjG4GgYUcVcZBHIosONUNbLgJH
4we+SBjGMEe4cONmho/v1Mgy41tqwr4RLB4GRO4fvpqodRLS3V1v28J
-----END CERTIFICATE-----
```
Fluffy’s private key for user certificate for example.com:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAoyxZD0m85OzTnvuZAuynNjq30x1Nwzq2rlc9X1UId4x+pOnw
aDEojMyVhnDipenbf2nIloBta9hvV9+wRT1mCm0NE44ibuDrzbJ7141qysBHgy
mJ152q3LbM84+uXpe+LWUJQF3v+p4GVN6fn4x+wjdbLy2wUq8sXBaOGP/VZNfc8W
3K0FYQsKysrsUexTbj0rAID+NRsGCmEtIAtE88z9Kw60oug14QUd4r4y/BGp6G
mnnhokxK+xBljTNDWb2Wj/jeaxwB3MstMYxsl1yO+yioAKrV634Br1BxH+G0V
i/slnhNERjTXJbZPc3UvtaUW/Cm2+Pdz7Ty1QIDAQABaoIBAH+b5vijiQ1r1Wnn
WYM78s4mpWeDr5chrvjmMQsyu/zQe11u455i1TFgcOIlIDQGtpFjLa7z5Ug4nGvYjvq
3QG6ieiL5mkfddDH2R+z13sWuMmYQG22Ta241VWdo+V/x8Ap+T9YhA2UGiwqSoA/3
R0PLN31TawsnE+hwiaGGsweujBvcaIJu4QRqGHRHaeEplU+tjfCHHE1fzuAmKYM
mcQf8IpduC1alyHe3Pyc00GnLyEVnv291xGQfW7ntqf7K0QDLA6+TvbG3fGEYIw
WK4DMraUb266Jlnj1XfAoxWOTsyZyV+KYh2cbwjBWAUSOSduAtfwa6b720nWd28J
8KYvrxECgyEAlcCJJZSavhlfqxsWC/WdQ8S3SimI62KSLrN3bIO/RO/60KUl2ap3
162zhNLq8t3DjpkWiZrukiX2odsU7k3z6q+qmm++P0UTwL73Bri0FimqUeVSYgAf
ZmFgG7wLAM29zhv0H2jgrwM1NSYj2tjyqpi01XqkbdbPBPxKrdcGqEYAW09f
4M2QKQBZfjcecPdpqwJQnh8cuoHS+2CNLYGjJlnjd/zAUGFV2+FPA1R1DmggAqJ9iw
15Yx3CbnknKbfh1lmHkccGyA+fjQaisq/Nzn3ya0FP9wa0tF0BoAht95XxFwXH6
YBUKrqoPF5Da4y27EL1nsIra/LtoPdqpphFzMCgYEA1gS0O0s2FA43uyTpeF3t
rmQpVillaB7KFSaigGBgU7Yp0kOF9dRwVT419sd48a7kb09ur2K08sHe2zSBenoB
Oj+hijNYHSTXRjNqNLuTP2fMU+uPDEFX/92n6WFjXKB+d1P8VSXjUKijcg36/H
1uhMzQZFBKXXVOPTRGOG3GdcGcYEAoFPmq8ZQZOA+BBnzqVi8GzuHuN8geFyE9JrSm
55Jpkt0HbXtst3txDjMBzG15KUbu9nbV1gB/PVBbcoSTV6vtDK0yq7Oa9agCyC
Zv5PArFh0v79NACsHixD2cIzdU7EjaFPN3u4APHFf7NsKhaG78yPPoqIUyp
O10XntscyGEXa1U1k1+5xwIrnC1Ut0qtg+6t0Tzc7qsoEopQTrktz/lsAXEhA6N
EUgWLMoNC1hP72V5tIVskxJgX08vg12eH11t5j2bSxMmbtSQQxVTF6sp38Fq1M
EtyXh7fDzJxKUrP7d0p2U4boJMF590noNnJrxj9VeSxEWURSK3YG/h8=
-----END RSA PRIVATE KEY-----
```
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pOMA0GCSqGSIb3DQEVBQUAMHAxCzAJBgNV
BAYTA1VTRMrwEQYDVQQIDAPDYWxp2m9ybm1mHREwDwYDVQQHDAHTYW4gSm9zZTEO
MAwGA1UEUCgwFtc2lwQxKTAanBgNVBAgMgFjBQc3Qg2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXMDTExMDIwNzEsMzIxNloYDzIxMTEwMTE0MTKzMjE3WjBWMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWcm5pYTERMA8GA1UEBExMU2FuIEpv
c2VxKjAMBgNVBAoTBXNXncgG1OQw8wDQYDVQQDEwZrdW1pa28wggEiMA0GCSqGSIb3
DQEBAQUAA4IBAwAwAgEAMCQGCCsGAQUFBwMCBEAwHgYDVR0RBEowSAYMGBh0BBYAMH
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAy+aHVXQN4BX/zlhiKmzGPCpe405FH3sVpKv206bcZcX9g/1c
1F/Plz1J9113Ez29NfQ6qPjJDl1+DTRqQ2cnqCK3Kr1lFyu1EJuAcs01WtdQox0
6nT01h8KOYW0VIT49FxtQBT2EUrVgX0UHB/AMNCpf0qeteJtZmPv8C+8H7zYp
gIIAwxzKpNa5g8hLNYVFZ1ddVBTUgNzQ9n1WV/NW6ig4N067R6H7C1NxxKAWR
ud05VRuW3FDgcgOR5whaRX9g0wBN7t9fe14akbSkq8ca+uEhaPQ5z9JMb+t5uB
3gGYXSiGwh4BmiqI+k/mQy5q3q6g6xUjwvW5WIDgAQABoIABAHCMxmgGgRS0xWLWB
P1kB0+1LSRg14+bqwbg6635HTAB1Yvzu+W2Bo20MnvMjRoeEoe04712J6boJzZvF
CKmKqryIYaKjkXgrBW/jT26xCWGFCNA1nplX1IWG5tDi4jg8AL0O04N7hyR0rrA4Rz
W0uvVQSYFFX4BhvdxZesyRwCqn3x0pSff95Ad+vuJd5CYuFZCuyGksz3fi+Nia
Gqs01Euyo1Ev72rsw2E5+wt83qX8B24XHrx+Yq99NhE81p2Cwd1UhlqIhI18kwWnniG
V3oLKliCvV+M6Zx/uzaMF0rdn5kEt+b5D0IksUAAa8LZsf95OvkLgw7aZaj5e
sXhAdGECgYEA8930YqU2+AcEkJcSfygw1M/X5k/1CcvZp0a8/in2hJW7iZg0hQAE
jjxuoIVXbxSwf9c2+z6g765wwv9ecmovLAqrbbhFaLfbZcsrLeEahQ7Gcu3w706px
NOEbfBF5mOK7qaQ1Sgqj0NF5z2PjsrxNCOnrMfFWdvCp/3Jp/I12EsqYEA1gu1
7/18hog1dmrOuvadnDR/nuXM9AE4LNRp09Hyx0B7vUt1ATbxt09ZNIlnH7bb
BTZ2R2RjzbSHXZ3fc0mgSx9Q3qa+xuPe14RcppHNjYxKpdP47nOlUQggCFL6kyU
ntEF+k6V1ZVsmGbB6wpHU1cjDAXUx71p6W49TEgYAMHpa7pExUDT076rH9tpCe
sume5441shtx0WObOAiPvCuqzr8dKm1WBJ8W7yOUs3ygh82JoPM81amqfwQmJZy9h
/5Y1AIW1jk++9Qv9nZJmM6OhDvDVFQme9VCE1hS/Mmox6rW8E5t7JhEvAzzy
Dqhtbh6wFW5WYM15zD3xewKBgQCRm1kY/QGFm0+Ih5ZMgB3e17GGLBl5Ne0nY1Ve
Dv0Epc3UQHQgQ7TCLDuY3yV190c8t7V+76JX1HDYy9u4bdBau/kkg3n+gd9Pj
U11xq8aam73rUJlX1WZ7H868rA16JqJ14tqPcW5S/rpr51n0Uyi/hXKT7psPizA08w
OV11qKQBdQzCYC/6wumGJUerVSYZd/H6+E3ntZm5zZ73c8+wV89oR7ZuzU0Y4
bYnRyF6s91KFCtLNGCRCEU2VzDkhUAgugq05rbzPudAA24wsRnc0UyW8LKhXHdckt
PVLs0vHrK2W/W2I1+p2exSPQP30y8T6fs92bNg/H4D160heHkuQ==
-----END RSA PRIVATE KEY-----
Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pPMA0GCSqGSIb3DQEBBQUAMHAxCzA fixture NV BAYTA1VTMRMwEQYDVQQIDApDYWxp2m9ybm1hMREwDwYDVQQHDAtTYW4gSm9zZTEO MAwGA1UECgwFc2lwaXQxKTanBgNVBAsMIFNpcGl0IFRlc3QgQ2VydG1maWNhdGUg QXV0aG9yaXR5MCAxMDAxNTkxMjE3MzJpMVUwTkMwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAQ

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

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3XQGAhDC5wQfvIy2JOeb1KNIN4WebYMSFlFhAjkix+oaMp4C5
v1LspspjR4St9nSFghZ2+TjZACNhQsIKIqDmp/Nc9wcWN96INngUzqR63TMk1CHsh
DSpzB20dJXW5905mdqfFqCmpskK+SReBiB2qKEJcfFSL5GGV0wT8EFDvgEW6
XoSXdsEgJcGSHYk91ViZPraaaJiITGfTCNh1htT0WVLa3RI1ZKD6U6fHfYdYWxKs
M3gAULoHBB05ArpJKWzheuJyb647AoOhaeDD3C1h6bLjs3Hips/az+62x+VxucKX
10280EdUcyUxvVOaILcmx/xp2EaJN3f2FpSwIDAQABoIiB9s231ni4Dk40Wm
u7w48acCFLlsSLMZqoMEKwCN6FO4zDT023LaqaJxee0UMuuKVXFEYWAP6r68CtIM
yHQLQMoOCdLN4x4y+d2tUEJERLq+9aAUu093ebDxcMntkfh6yNyUS/mk/KQMbpFRT
1dn0wXsJc19I6yxA0rkB7/9UEcdUt6vzdbz+aqXpH2H4Tje5OWZQXkHzsYoM8Y8
c2Xwud1zdqtVorrOiirexopQf4CBQnBxOGmbae9Wf27Kw2zbM5+b1ZFgdoNxo
6Q3rJ9EDyWyVRMAq9a67a59wST1ymyCOc6FmFoTCmG1g0MPHcEdvUNYPwd2322oK
ZdfsawECtyYEAEwMlTdhAE+9T1d2q11LQV+8bdTHQ9rsQ9S9SF+i5ShOpA79E
asuDuqKU+TiewSo1ircrIyzQmCc1fnfBJh5y6GupkUK8HDLLkA29fV3ZJe+Y4ZbL
b4TEy/RxEECYRgtnqIw80o11tdobNwwxVsi3mtrhtOfpfPBERZUSSaCqYE5A5JG2
aGRcKzyAZSGAznZmqxCQ/p1MnU+t.Jb20CqO6/3gsxi/191LwtRfhGx/pTycgZWlpb
+mpnDqexKtwoldbjorrUADw84zG4u9d+uWOCXEcpCVIEu4DZsRURdyz0z1klvJaUm
NlQbI1j8kJUFrXTI4Rzx1Xysf66ndwAxDPDdI+GECyEAYoYFrYY+dohSvs9uijY4e
FV5n5t8E7iQF7L72SoLDHy1DjOv2+VF71erbDusJ751q9hj1q7id3ips/M87P
2qJsMTGboJrSToTsL6mx16LCD5Fmnj/yfF1beaM9FPn NgT4ip388SyPrh7qv7kp
3Ao7AtXtwtVzBPVucz8A/8EcgyEAYw2ps2F13qdql3ns01Ho3gqVoaGUUUI0K2MI
wjYM1/AkZrR4PKthm1PEP/F/tppsBz2yBB06XoYyay5+10DwZ0yGHJNjeR7GqRh
hqC0EHQQuzkRd9hur5g1i0oMqCF4tBV+wL7+YnKOAUiD3g7igIUa6fjxew9o
FzBx6ECyEAYAhvSeqwmndotdpNgR3Fk1CmtN7ZFPnf2rsuBaoYnWt9U0te+
+bdn4fUB9tQo2U9cRctt3zw1JKHEPjm/0FxtQO19oE6e69jOwXJKNeSsa
GjGUzf3Vm2bAeNM77s8HC5MQ9nksDuCzd1VAB2bMpm23oIP1boE=
-----END RSA PRIVATE KEY-----
```
Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pQA0GCSqGSIb3DQBQQMAEAwGA1UEAwwDMQgD
QFwSB1lwaXQxKTAnBgNVBAoTBWQgZ2NyZXQgUmVyc2V0IENsZXRlIGF1dGhvcGF0
ZXIgQ2VydGlmaWNhdGlvbiBNQ0ExMCsGA1UEAwwDMQgDQFwSB1lwaXQxKTAwDQYJK
0AgEgwDBQMCMBxGCSqGSIb3DQEJMA8GA1UECgwNyZ0YW5nZSBDaGFzayBDYWxhY2N
bIGZ1bmN0aW9uIENsaWVudCBDb3JtYXQgQ2VydGlmaWNhdGlvbiBNQ0ExMSowEwYDV
QDAQH/MA8GA1UEBwIBQ1NpcGl0MB0GCSqGSIb3DQEJMB0GA1UEAwwCyZlZSBDYWxh
Y2NhcyBCaXQgQ2VydGlmaWNhdGlvbiBDYWxhY2NhbFBoZWNob3N0MDQGCSqGSIb3
DQEJMB0GA1UEAwwCyZlZSBDYWxhY2NhcyBCaXQgQ2VydGlmaWNhdGlvbiBDYWxh
Y2NhcyBDYWxhY2NhcyBDYWxhY2NhcyBDYWxhY2NhcyBCaXQgQ2VydGlmaWNhdGlv
biBDYWxhY2NhcyBDYWxhY2NhcyBEaXQgQ2FsaWZvcm0gQXVzdCBjcmwgQ2FsaWZv
igmJSb2FkLmNybCBDYWxhY2NhcyBDYWxhY2NhcyBDYWxhY2NhcyBDYWxhY2Nhcy
BCaXQgQ2FsaWZvcm0gQXVzdCBjcmwgQ2FsaWZvigmJSb2FkLmNybCBEYXV0aDov
MA0GCSqGSIb3DQEJDARYFhIwIwYDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
-----END CERTIFICATE-----
Private key for domain certificate for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA7Cx2A9VI8zB4tawEMdRJclLPWTqm7ogxWDwhYx/WDRNZM9zm
NQBCVaUs22Y31xBoLrnFV5y9/essyiY9k9lp2t6S1NODUpnXbBCDI/6/H8d/UEHKP
ezYMT8pmqGdCMVBrAccxz6QKlJrp0IVOPY2cDlXgNXE2YoYrF/7kdjF70HIjo1kY
2VlHwAM9QtwWdpRyE7Tom+sYW3Wf20C0x/kLfx5hgampcGWpk5XxaqJaqpWWd3gu
u8NmTCO1D+jS4V4uIfg+aHdy6GOSQxtU4g+BW/uiGNY8OxpKBOQ7TzccGF03g
xc8rJ66dfmwhHnZ83K037aozuxjha99NmUavHQ1DAQABaoIBABfBYR2BiHpi0S6
ylE6asjWrILhLd76NFxrr/Aig79MbUCeJCNio2N5+ckXvvx219N0U0+tt2ti3L
KGyfKec06iisncjxKgn0nzv/o3n01z97Xpxb9mL9t3GHOYRoUvK6xGpGIIo60B1Cz
F+8pk0jegc7eVF0upMULHm/FcMyY3ON5cvChcAE/nCcW49b2mH3qG+cmr5UCkdUY
baJyLd8QtIf+umtrfYzrR75c+4wrmBUjv3w9poMJmJuEo4slRaDnyeKJPSNR/6/LJk
tqnggNif9cJ9WqF6hWA23Ddmmu/ksRtn1KOZ5xMv9JboF4u6Fvnm/m/h5Og4CP9
h2UWIQCegYEA+nV2pzsCFS7jSebVnvjChqvdj0nJAi1sqCmriQ1IT5PRm+Qo6u4UT
PVN4EGOms8TTJyYvKpoaqQ36VLw/Wr0jUum+z+dv1T1IFWtas8RNmzZHMv0LVfeE
Qu2fT1612d/L9GMUcYa/sucX5EG9q3LC+Qo9j9lw8ehWjQ2szSWEYR4dsQgYE4AWXX
AqDdKjHrqu2h248gZsuogizQ05iuXhXk2VTPqo192Mu8m1Htak+eov3/w0jzxu
TAQbf/tEFE7LIGjKqAu7mgG/aN86BGw4dpBAPUZDL+DuKfBdZTB/ZLiuW0p
40RC0Up5nTU9wzIKB7a6n5S5R0KXX1iGUphfcCgCyA61YdPmziU0fxJ792r8UqV
82KwBwqXqpyLsvqzehstSaRs45a9S2qlyLlveIEChm2S3i0lRSU/r0ncPvEJe3g
+SG7Zkbh164p3WuUbGdmHGCnsN0+3CTM/jagGlzmzbWnV7+MwTN7TIV3Vh6U
EuUkU1biHsXv53zAbWeHKwKBgB5hWfLCExsAB62o9dNhImP2y28YuClR4L4tJReyu
MNz6AIyayahZiTXbGo8f9EaDrxzvYxZilMKi1EnFrq/Wf48O3cpMSnInk1l1kPO
kw3qOqIdrods3jyzaTzC2ctjYxXKUXOFdjtW+baA9kzj9v3HCGc2qbl/Srtraio
JMMFAmOGAHb+k+C4e8Ww+JXbb/DgAkOKK5vZLHeWBIg9beE1626vN/oFEQXp
zqwYno6zQaofmS6an16P27N1CGJxh27eBT1LPlNCX1GTQAtEkmXtvnAZNzXC
5U00wv5BkLX0nbhJW8N8BwzJhYup0U3pn99Gcf+vkj5Eg7Zfge=
-----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+jCCAuKgAwIBAgIJAJajhBdO74pRMA0GCSqGSIb3DQEBBQUAMHAx+CzAJBgNV
BAYTA1VTMRwE4YDVQQIDADBwMDAdBgNVBAgTAS5lcG9yaXR5MB4XDTExMDIw
NzE5MzIxOFoYDzIwMTQwkkJjMS0wDQYJKoZIhvcNAQEFBQADggEBAKL9
wUGRhCqdhjzy4Bx0R5kwz+NHvbsb8r1PqfdcbNuBCw+rD+/uxxOG3Hw+Mraj5
U2tUehwz87k6SdgADzL/CP2mzjcJo5udHi+tzjeg6zklTSZYQrL3F8v/AqcUfF
9HPCgkix/htaoEMy2znJ0d7FME9w7wb3GxxqWTUz19+toCYxMLeQo/jwعاد
4ybn1P3CW05MdY5u4ncCsRrz5tLYtAVFANLBFefFV+S87AwrrdeITT+iyB7H
Jjt+24U1MC8MtctcHB1FPbuVc2kmmNEQuTuZelCslxdXyY2+kn8ItnLdv1mvLpXA2
2Y41PLCSj9A1qlqZ9I=
-----END CERTIFICATE-----
Fluffy’s private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAulcjiD+lADV8u+uih4mA8cj qwBBW6Cyb+khLcFFkKGUt4ULq xYnTruIJA3rK5c2jMLcu5Z1lsWxcVChWylhjHAdoa3syfuq8CTyv2xeWA4olbVa jEXeZb/zYusc73xlMlp897fqqI4GpF04y52Sj9k5tT+e+u1lKRGxVbA3mLVoVhtX mwwEem3+IXW2tiug55pMU+Ua+qGCRyA6KG75m5NvRaUDehBgQYDxKUX1IK46CUf 6wQ3wT8SVvjjXRVXc5Iy00TzXb1IK21oE4ydfLVyeCobYbWmwy2N5mD/+WLp9jzF 91EHDyHPwuglwN8FGCt6DtRuguaVX/SjDJTQfIAQABaoIBAbIBL1i+B95e71yw /MOxOrKrMrwfw8ElftppGTXhfjN31MbFIFAO5hJd3GncdwAM1lYks6YEZ+mu/rMHz wp2FXC01fGSeb78tCMilb027v0fZXUkTnxR4aj41YOHyrLg7yfrXj8ER8WQ1KPMK PVKmL0wp34+zjOhqUdpRs3xhcJC1Q81fc1hKe2JoixNDoPdfM3azt9q8QvLQO2I mjww1H1677G5o/6cMlcmO0Fepovj/3cUWimRmvPv4eyGHdNtuFXKfpB4DQQMQL7T8D Fo0HBymHIOzSSE+gYgBFOb0YNQn2CqZrfE9fd0rRotrbXf6tM+mak1xeHhKhFkaa JP2osbUCygYE4MaetKsa7azEYMc4TkoXhVhV5H161j1xR/6h++yF00IOOB3M9yU3 5n6vLpypgHbW2bKO801WFO0F4syvyKYR2elmUDraH29DKatRLEKU9K82RG4AmXmK G6ZwWOx6JF35OnAKVj/7aN9jc4K1v6EFyQGYEXbp4If0hFfbJBAe28CqYEAI1mx iKJD+jW9y9hpH51lY3j+a5qPNNvmgKQQj3E6v+R1xW0wxoCBOYRhwhHBRAB/ Sxh93PZ8rEcnKhxp6aC087ZGc0l15UkH+rwfd/3+SahPrguaDiNg1kcu8VUrP 8uP2CqJoDbbi5UY2UR97GK98b8k2S66kDT32mQMCgYB/KH3R8V7yj01KcqTc1UWl J1E3/gB4S+wQ8EVLth0FVCPOPsdLz1ItFw70fUra01k/SHeSfijDIdghN6m odFMQF+7vh47zUwurZPC95n4nk51hTkNj1V9e1JTudjLuc3WP3yFc2JUX1kObE+n k6zuufCoUWFScij2jubgwKBgC7lRHe1JjkJd2F2nX8r7D88y/W9wXVtDWqgiE4X XQ/O0F8A6ijBKTa5qScpZ2BAbdpIPjc7VE4ta1A8FvQPXVZXcR52AFAAh4a413C3r V5oWYb1lI7T7XA2yvUO8W/Gnds00zU11T8G6W+sAYorym/M8k/t0Knl5H1feEyEeq Y/w3A0qASjoC9Fjy2aBwH85Qainm/Xr3OhF4myOGWtHxrXmzzoO2YdcoOIdzrl A/sQRv0fHRwyoak2k2ALprEgyxCdMms1h9y1AcrefW23RqC39Dy9RrRkwa ArJmcEdRESOSIYHhXGfElQM1wjjUXMWNeYcLtqKWKsLDDTYyFQE=
-----END RSA PRIVATE KEY-----
Kumiko’s user certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHAXaCzA0BgNV
BAYTA1VTMRRmQYDVQQIDApDYWxp2m9ybnh1mRREwDwYDVQQHDATY4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBA8SMIFNpcGl0IFRldGVyY29tIHN0aW1lZGlj
MQswCgYDVQQLQjEyMzElMTEyNTUyMTJfMjQwM0YxMjIwMDAxMk0xMIIBIjANBgkq
CwYDVR0PBBczBScxIy4xMDkwMDc4MjY2YMowCQYDVQQGEwJVUzETMBEGA1UECBMK
Q2FsaWZrQ2VuY2YwDQYDVQQHEwZrdW1pa28wggEiMA0GCSqGSIb3DQEBAUAA4IAAL
bq3OKM2Kojma5I01wpIa0RXYQ0K//oX/+/j2eakhfFty/R9yer0KaXWdLr6dtncIsu
z9rFhlTB91Hg6vN009L+xonbcs7siXbj3qZdhb7oipI4Po1XVetyu+SzAVe6MsU
5lwLmpQpIzQdssJyxaAsW+AsyxunhWWiP3888UM4vXjcx2uj8Gv78xJgjij1lQvV
s8ojWMKnAGLaj7grTbmGQ9o0e+cq7hWuoGB1QA0R0h8zWQz5AgMBAAGjg4wgasw
UYQDYR0RBEwCyIYwc21lomt1bW1rb0BlcGFtcGxlLm5ld1dIYvaw06a3vTaWtvQG4
YW1wbGUmbV0hhdwcmVz1bW1rb0BlcGFtcGxlLm5ld1dAQBhNVHRMEAhAAMBOG
A1U0dQwBBR6WwH61U7BIWeikM35fMAEi9xazAfBgnVHSMEDGAWgBSVRX5fK+p1
mBKRPBNjx2iaW82Jz3JzALBcGVHQ8EBAMCBeAwDQYJKoZIhvcNAQEFBQAdgEAKE8
y9YyoZ1kFw4WxaPok087s8EweKbfh4TuYQf5YcSIPw00o2Gj/qGNN1juiYhE93G
F+si/h/MJOM6cc7SLB5Spq06Tt3PyPB10Z0Wk9kah92kD13axSr6I9Plsyp+Xsr1
bzoZ2y8nyj/Yzrk/q0aHq5Q5NP05qg5wF+ns2t+5Z1B8LIi5nkBqOFQ0EU0RIkcdF
CUFRUoj026GxAILR6wU0Th0fzf55Azwi5Y9Y9QmEjFhkJbYLLs00HxcJdnt+6Sdm/vN
MeMJdzp1lx+8pPhj3goyz7nAwhgzc9RT33ra33BWrKMQ6esRLQONJ+2RslRlHP
O7+kxVxmj9AAsA291wY=
-----END CERTIFICATE-----
Kumiko’s private key for user certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxP0FTe58Qw7ual+m9JHjyBR/dyoTWx0mgorIg4u0n1AYqr/X
YEYDwV336yCFUHSJqAgA3iw7rB3bAVHhxVr4PqU2SlRA4MHHHdtBeQFW6tzCtjY5mu
SNJcKSGtEV2ENCv/6F//oX0mp1Rbcv0fncnq9Cml1n0XeirZ3CEros/axYZUwEfZ
4orzSVdf16J23LO7I2496mXYW+6iqSOD6EjV1XrcrvkswFXuJLF0ZcCS5qUKSM0
HUrCCswFvglMsbp4V1o1j2FFPDOL142nGbo/BryfMN144CyPULibPK1jCpwBi
2ke4K0wZhPdhuvnIO4VqBgZUANEIfM1kM+QIDAQABa0IBADuLR+kwp3sv1rcX
Z41Fs0mBANLNeKpA4+KJ/JCR7xQ9bfACXhec2AnuWL26TUNRFgoKlZdvEookYE
gHD57n36dcf9kr7rpH5xio0RlJncoIFNFRvPNZiCZBwNIAAXFLnHGtznVnpwT7xi
axMNqsr6e1iO0/quAPk0u5x6e0+j+j3zauIFd1w2R6moBMUaTATau2EYyLuC9A
6bfZ2AFDchPVlw3jnMu0tAJc8Fss8Xkl9HUXGS2UeUFhxWfkgCgwChW60obGmas
E7GS7h49qVqVq4hG5SYV9/MmQ88GmT0LynOyzFBCpuwqOQTHw0D6741dMSL4kXYV
jcTNakkCgYEA4bJN2ILIs3uWTJvTNhrnnWn1QoZBDkhe1LuNso1Xu0J7CckkUvs
nqQYOznK/9N8usU2a3csXHyppuGrJwAVf648RSPDUUQ2XoPSL9NeuZ5t5V1fT
1VvWwAmCk5zzrISNVP7pU8DtcLRh4s7/M+eUQBO1ogz7fhyJhFsvQgCyEA32me
61N67aHkqMLaO6ZJ9IK/3SfFIPpjw2ftk+sQCqEazvPvkT7qFz2+U81Vc0XXKJZL
axexopsUCLGS986TEA0poYtj9k1p62Zj8mgRZLU55g+gRdpAfXmQg1ctU7U6cDIw
Spla6tXj9dC/Gf6ylfas9VBHc/80C714yjldCsGyEAgA7gyuM/CSY3MrArBW8f
4W9gkIgtwumn2gjobtjek8GKoxce4Q9qa0yi6HP2M8h60gkUICzycXZGCF
s33Yxzc+9rZTqzuX3bl4AftNF150P0FOEn28KhXSrmbXbwG+LMNjNUPF6yisuSw+
cuQxAliyey0gs63hPl010CqEAYEuEcILGQpTGYwC93n5Vu6ir+IX08gsyl
ewlihrakhliWITYsxgyHgwKB4101W0EHNW77DVPdPcs3C1e2hzN8Wm7KtA7jHo
YZfemsFU99luPwQKMwQ91QoKer7cOHdBsRWM15q4s5ujKnmmKeShtHj1FNgXe
q4k40wCgyBoATsL1LMltq7?sxhZSMFbLP1/1mGWZDDMAzDW+XXswv86brcWY
8c3hbohuJkpyAzba4QorgZG+gtRmodLca+tQFMb0ETHgF1NCY+WohR5RNImbCS8w
dsszgHWF1nrxBL1DF1HZwSqb2tLbyBp1Hj+fT1Pno6UTx8aDq4Pw==
-----END RSA PRIVATE KEY-----
Domain certificate for example.com:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYwxp2m9ybnhlhMREwDwYDVQQHDAYW4gSm9zZTEO
MAwGA1UECgwFdc2lwaQxQTAnBgNVBAcMIFNpcGl0MBcGCCsGAQUFBzAChjAkKmAw
DIGQIDApJ胜利LvYQyYoVzUyZGQwMTA5MDhjZDQ4OTEyZjI1Y2NjNThkZTFjZTVm
MDc0YzY5MTAxMjFkOTRjMDIwZDQ4MzY5MDY5MDc0YzY5
-----END CERTIFICATE-----

Jennings, et al. Informational
Private key for domain certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAoRW5jJmVqp+qSz273JZBGn1Z8j5WxCaakkfe8B9q0SUJ3GUFXv5HYJpD8adcOCtSUALlKIPy88erk4vHWCmAMXxa2ZxcG/ArmXM37sxQLL8whIyvsTY136nLw3dGeNYMl5SWMChgCm7hkFL8E6Lx0jMARG885+kJtQQAEXlqPt77gO2QSt2QzBxJkjncgc/fAM/GYzMKw7D9+Lxt/L2PuaJQO7PTUXz2P86nPO5rYUfGBPUroZGW50bglrBKIfVbVb2FpX3+pU+1F/HmB4H37a7os2zd2d2Xsz10dkxcxk2Kc3gKxyi1hJBndl3W1Fsa2r6u+ySKKcO55Q1DAQABAOIBAB19ig12A0ed2zXJYCja/ON4EBbRdh0uumO necIf/6J3xjTD2Nnt8T0gdJUJpDhjjwZWQzz7KYdzDN4j6AkeszB30sT2MTFob/wiCt6cAH1VrrKZ3cK6zyY217aPj1H8IUaUr1T73Unt/DMP6mgFbo+xQZ18eVFc8zubc+BK7ksN4N6b/zhmwh+FXIyiy2EGDN1F04TMhxPD4w8BIMU8oLlE8A6GKmxA3gMuIis6Ruu2hpGkjkkhkAx/yzu1s8BcMoLdjdyyH19RiSr n0VFfe0gMo2PdZ/94ynFFDmBXTqBBabT09eycUL1LOg/ERmj6jIImGSHWEDGzlzX0UCgYEAOFDUek2uLHYltXw1lz7T1dytItiYzQ/MeXaq2eA96zhJ1JDax+55PQixEEfhqFJF4e4cJkXQSD7aixy7j/pkFGowFR1B4pwbLDuHnIYXsa8Kv0OpPJ4MTAGue4QFz2d524KH755Ub7trCEiDnqj4j4DA3gPnjQkX7dpyVcCgXEAxfUX/zMXgTphXW/HQD7xXEs4Fp1xjzL5BaHoJnM7WhmkWvUvMaEE/19Rgpy1XrINjX6Kb2z9Uh/g/BA/cyYa3mdIImTa0/1uei9kN7t11pzwIUAdF+FyQ4tULWwr5cgrzvPjGg9tXMthuiBiLSumVEwvC+P6kail14xpl3ez18CgYEAEF51sk2clqM1qapusxJxmLJbdaA+w6yCD9m1uqagXGO8usmqgC70KrphmE0QFvijPNuZ71WZ1/1AKvcdz.kEdDFU5FzxT4jD13zBg713FrXQ5Xb+3YVfNb0vcWSc1VNjcr8aMi3mE8mUfhtFnRQPOnu6nqmyQVjnTkCgYB/3z1nknBQk9ooZEU3Iq6TXL5pLemOloFQcjcKkjVvNTRcXM5Pn9PSEaipL60Q3+sOvYg1nyVOSwlPw/VVBdfh3rHxW6C6vKeucDz5Jy6Fgw5miJlbxy/WTaachoI0J1W1i5j1igHcSc9u0hOv9sbqJrYse6+HV4sNuo h0l1chQKBQcKLEH7wQX8fkw+yRknmaA0z5ZH1HUQw/WScoCOVnoy+vwovcuiUTtcbW1VkteEzJ0uFy4EsF35IIZkmsU2GGrKcpx/3uq2JfMVopJzJN9biFM4uc1ckq9f1ehiVIFVmnq+dVnXBgCknhYR1mt9b3B6mDqrErQjK1TlrqyA2jQpQ==
-----END RSA PRIVATE KEY-----
Domain certificate for example.net:

-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pUMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDjTYW4gSm9zZTEO
MAwGA1UECgwFc21wX5QxKTAnBGNVBAsMIFNpcGl0IFRlc3QgV2VydGlmaW5NdGfU
GVQV0aG9yaXR5MACXDTExMDIwNzE5MzIxOVYtMTEwMTE0NTkzMjE5WjBbMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMMDQgMBAGA1UEBxMIU2FvMjAxNi5jcmwGA1UE
CwMEREcGA1UdEQAQIBMDUwMDowMDgwMDAwIjAaBgNVHSMEcDAIBMjAxGzAsBgNV
HAwTDQYJKoZIhvcNAQEFBQcEggEIBAQCg0IBgNVHQ4EFgQJg9c1bXyj9yQV4u7K
bAe5v9jX12Ph3J9s2c31p2F0EmaVYKHtxCgYIKoZIhvcNAQEFBQcEggEIBAQCg0
IGzgTANBgkqhkiG9w0BAQUFAAOCAQEAJry8LukecUV4DU5u/s61lymQDLPeNv94yr1K/crW72JtF+t5
6zFO/1p+1VxjRnFRnR8s9XH+2yU8PZsuFZ4UvlgGflGkxK1x3/22LPz/fW
-----END CERTIFICATE-----
Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAqhbHyDUpucZfZgQ5etuh3altHR+mrhZxS7+0hYd12x4AuWq
0O8pxYiIrAAVtsna5/tL3k6wGzSEBkDcG6B+1Q1LVuMOGwpWqJ2A8sxq+30kvRvN6H8
DAGNzTA0HP/qR2Reuvb58XDDDNoKgrD7xCbhufYyds2paWLa1o5edi3gsabLon17
ri9K9dWp7TN7l1x8opykFu5X9XMxv61ucfbnMTQL+jVv7H9V93DIOwq0Q34DvKwsj
Y12ZVSUCfiOxa/Hyy3B1tQn1fJd7MWbDjHOtQ1aRvSc1c7QRE17uEicsK/2nr82814
exKw4a0pJAgLqTvzgIPsPmvwW2ctWAn17s7G3bwI5DQAAbO1BAHIjpv+B5YV1TL59
+UCrf4jyKVLGliOQf/Cygafjt3ZTVa6v/aRn8Rkgb8Xyjr9xsvZVB1tiUbdM4Z91
8faVSKLAWs:jthtkfSojTmzU77x+iDcG6LxSzekAQgAIJ7sRL+iEzl/Fm1WlgEYh1
GIW1lGgHOln300eCy72dwvAM+2haz8ne8BggkWxM0n0b1Rc9pVhOFCo+jy11HasJL
0OBh5111bmZ4PuUuUY072j2j665gPm7i0n25iegef842JkbqvA8r0nQ12q7YtELW
6QyLt0voode0rH28IEzahWAdmIPGCIUcFM7RmyInOatQAoDVEU3uYnkuUQQVo1/JT
Xx46CcmBEcGycEA4cd1v/IVx2zPdWio/0Ma9j4zqef7Pn55RMknjCcs5xVHSMINw1U1
BcYozx77vWb1uXIXOx2q9emG92as3+vNxBoeu6EBQ/fK16cQQOH52nXdrVlsgknN
5B5eIFKcZ3KpFNVrWq0BC6csDndTchp95T1ksWxesLcz3Vz5UXMmocCgQYeAENVV
+SaC1QJG1T82Z3fkyE2hnHqRUFkKc/tWQJop5gnE4w3Lq13SnycUQr/sDylxQDE3
6Co197Jcz27ggDq7gri1xMznRILMeG7bb7fFwPE/SKV0H5uagEB7ktFl8x1JkJk
yOckl1ul1lQjToSs4uHFLRQKCDSEpRis7W2rQxyEAEAdKBBXya/nykYDUqDii57
1PbFkDD9G5x+EYPTUOx6yjgpbF3eVANhVq0oDTRdTmjrY8Tdpx22W1s3SbAS7W
hfcCtCwveCM+S+D29onKoQuv761M6C72j36esEXBuUPea072RZDCx1dsmEjInN
+M2KxNcGXi91tIehJ31fouyCgYB9AUs1PawaeTVX13OduyuhQ0xO0nnMa491EuB8
FpCipDzUhmyZwvJeLPxFwQLg1mJL2JeNeRfnPqDr152zqXKs9z5j/wBeN12BM
ctXZLp6vphnHjg+xno4eQSHzKvYvb1hhs51CuDx+pO4sNWEexM+Qd2FCXySAS
UCJ4QSBkASr85yQxQhdBRAAM9JZLGUpPnnTNV98f3EfNnX7zEZnTasnnv18v
65x04h5c8ohcJJKnXqel6kk3c3MWpzwPzs1ha32MEoJPCgwb8xLZr13YQin6yf
+baMFBdhm1gOPB36ODY4BikcwxKzQ0n3XAT1rl7NRRV5whr2ejkY
-----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 as a
signing official.

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=Sit 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):
00:df:2a:77:93:11:5e:75:14:d2:88:54:bd:16:50:
d4:41:3f:7e:2a:e4:26:d5:a3:33:b0:5e:37:1d:e5:
f9:0f:dc:9a:45:0f:28:8d:dd:fa:15:56:d5:35:17:
77:8d:6c:ff:32:33:f0:18:87:c8:cc:5b:54:5d:dd:
65:ff:15:5c:f5:c8:ff:3c:54:8a:b6:7b:6f:f8:55:f8:
d8:df:af:9b:40:45:4c:92:0f:aa:ab:2c:ac:8a:64:
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:ca:da:1b:52:ff: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:33:e3: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:
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

X509v3 Extended Key Usage:
- 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-----
MIIDzzCCAregAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTA1VTMRmxEQYDVQQIDApDYWxpMD29ybm1hMREwDwYQQDAjMTA9MQswCQYDVQQL
EwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMLZ2FuIEpvNzE0MC8GA1
UEUxNzUxMjAwGzIKMQswDQYDVQQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTEM
MTE0MTYwNjAxMzIyMDExM2YxMDQxNjMwMjAxMTEwMjAwMjAxCzAJBgNVBAYTAlVTMR
MwEQYDVQQIDApDYWxpMD29ybm1hMREwDwYQQDAjMTA9MQswCQYDVQQIEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMLZ2FuIEpvNzE0MC8GA1UEUxNzUxMjA
wGzIKMQswDQYDVQQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTEMMTMwMQswCQYD
QQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTERMA8GA1UEBxMLZ2FuIEpvNzE0MC8
GA1UEUxNzUxMjAwGzIKMQswDQYDVQQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTE
MTMwMQswCQYDVQQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTERMA8GA1UEBxMLZ2
FuIEpvNzE0MC8GA1UEUxNzUxMjAwGzIKMQswDQYDVQQIEwJVUzETMBEGA1UECBUKQ2
FsaWZvcm5pYTEMMTMwMQswCQYDVQQIEwJVUzETMBEGA1UECBUKQ2FsaWZvcm5pYTER
MA8GA1UEBxMLZ2FuIEpvNzE0MC8GA1UEUxNzUxMjAwGzIKMQswDQYDVQQIEwJVUzE
-----END CERTIFICATE-----
Private key for CA for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA1EZlUfiEHLWTR6UVFABS3cp3xxFedRTSISF9FDdT9q+KuQm
laMzrF43HeWNXwcaYc7k/0iEnf00BepYmHM1i8BhSAXF0r12Q+Bzhc18XWGSa8
fQv4317S9yaRQo8qjo36FVbVNRcogNL8H9aVLUI0LeCq4U639D1t9o0Mzg10VNPi
2NyGqTUmBrISVAhS5T9KdpV3jcbQM/AyH8j8Wl0RD3WXXcXPX19IDZeU279bh+NjY
3617tQVMKg+qsiyhqGTVmSiEHHi2Q20RZD8DFKKS2GElrBpGmujb7JzknC3rVeW
K4cV2cIRA7CC1CPA91E9PYc1vJtDP156oaO5QIDABABAoIADp7/7pIH79vnc3
z7hGNE50aGBHuPrSh3yJG4a+O67XbzaRW2I3XzuaiEHiixoY7duha97Kxu4dBjC
f2JijR4uAIs4aV7NdD09VNW3o8NnKWWLEnV288Eo2Tqgc8wXz/B1eL9nCJWcH4Y
Jw1rK9KWmTdqPvBdCwcP1I9uZduXqd2FBrsl6t0Z+F3kdvUwYAVhUuBS9sf4Xib
5GA2CFLp433giO3yr9KigpcLvbAhMIPTX76i65m9xGGCcjhxP/dor0OHcczRD
y0WFcBarNkJug9kEvun3uGLaVFonU7RqcblFXG07ea7G+mpf3Cfn744kvFEXz04k
8WLW6gECgYEA91K9umHwUeb1+xPB42a540vVFC7nL8e87e/aTNcYMI013uxyPDPj
TNEfgaRobptmwd2HVTxj1Q54Fe+pE+qS8DOORm2VFw91iz148Wm/6j5P+qiXY
tcZDPF22msW7uAqyaOHeFhmxo1Bb0UH5q5Yrca5DmmQtaxcIZ+IECgYEAJ07
6DamIgy0eJ02GKHU/Hy8RVq2gauzCtmgmLQrZw0mz9hORe1a71QU5F6Y3HQRcTD
RDDdJua98Bj0WTKasbRgxjmHqlF4pUd7ctWFxsbCNFTosgPH+O/ZPEH4DK1O
rb1dUzHPuD2o2Q72KtSPMk+iKny21C29cm2mKmUCyEAsGoX4fJ/HpDMzrKf4gTG
CoBojxZ4+xBPvTF/v0L6t8wTCG3VruGp25gY4i41RwpFEQmuwW9cnE+N2TJQkXQ+
47VPylv6r/OSAM995csWw2ZBF6w42v0qFR3W37AaTUCgGFThKBq7+jhQX/FqAhO2c
6KxxsM5vfqToj7xP7vycp5ECgYA4TqyWpHcpq99QV4sJUuM4v+dbJf6dQ96qNf
HEUgNc2BC5NWx7D4+rXm7qWNCt3S7N9mKLU0RtBGec2RwVFOuj7y71oXmiuE
BWNfOqjsS71Hv3aI0NW/Ezqje9T7Ov1XFg1Uqgb4+VoaZHYyE85G8z7pjcCxed7
qD7L/QKbqGeCLxKs519/EqwW8HKn5q0/51/0T0u3MCclzCJfa2BHMasvR4Ld-
unmMAMDEA1FPHOFS7fSmCfsP8NY7+W15/k9MuwpwQFST2Y8DrSDR6PFp1FRt8u25yX2
mdRbU3VJsiAgEPEKpb01XpLeLVoTFHFSazgmcP1KKxq0wL+0w==
-----END RSA PRIVATE KEY-----
Robert’s certificate:

```
-----BEGIN CERTIFICATE-----
MIIEJjCCAw6gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMH0xCzAJAjBgNV
BAYTA1VTMRMwEQYDVQQIEwDPYWxpz9ybnhMREwDwYDVQQHEwhTYW4gSm9zZTEO
MAwGA1UEChMFc2lwaXQxIDAeBgNVBAsTF1Rlc3QgQ0EgZm9yIGV4YWhhNjIwMjE0
MRQwEgYDVQQDEwtleGFtcGxlLm5ldDAgFw0xMTAyMDcyMDIxMTNaGzIwWzElMTE0
NDIwMjExMlowVjElMAkGA1UEBhMCMCVMEMswGCSqGSIb3DQEJMA0GCSqGSIb3DQEJ
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA09wUaWtxCSwLD52VCMFk
Igbvn5wwBja56xQ2hnnMQ2x/z/hTW6UNduyXuhYQn+1XftfS5+P0J8OFac+iMTk
7zXRY5EgaBj0jju0DwM+oADWwybnV4hkqN6LSFESNSuBuTo3Nbj0bPySyYPP9SZ
Y+R+FAqycxfx00E262pwvmdPCgtK20QC6LtQ2iyYLEWEfgEOncOpA7coFSjdzaKt
qw2f/2n07Lp/S72bKtwNh+Jm0SQ05Xe+iHJdkw3NDq82Zw29ShgAS1c9B56FRk0
qRzPogzcd/mBZ9xQCo152fNw7/pqiAdGz40qrzuuy2UYwhvA2ydCVqtMCLM75
IwIDAQABo4HNMHkmFEGAiUDeEQRKMBGnFpDpyb2fJcnRAZautBw8ZS5uzXSG
FW1tOnJvyDdEB1eGFCtxy0M5ld1yIXhJlczpyb2fJcnRAZxUgbxBw8ZS5uzXQw
CQYDVR0JTlAIAADAgAaDGy3OHk5Qw4Bw4wYDYVR0j
BBgwFoAucnDPZha4jptJbpcP0hvpy4l1v6cMwCwYDVQR0PBQDACgkMB0GA1UdQJ
MBQGCCsGARGUFwBEgMRgBFgEqCDBFzANBkg5hkiG9w0BAQUFAAOCAQEAJZnpGh6W
bu6xnFpD+h6pG3Itu51J0eF/cTlIrLKgN3JpdtQcaL2a40/JpGdxbdfj0d1DzdL
1eC4A0F5P2/5/4Edzdi0R5sdoov1y7j8zcBBo50er5qJZb5pJowUXZwmtc1jUuM61
izF0cLMPXASQ2i22dV7wdjo2889gOSNL365K6Kenx5v0E5u9/CmYTuem0t4a4Q3
rZM5DZ9W9hN/1YoCnWN/2981Q5XgKZLnx7ubVQ7ywFXB1IBgADYESgyBELE7
7owYGomaKlCwzjtcT1h8D+AM5DyYFNI+3plyY8fo+hiQkL1rdtgKcYPPGaeis4
28xU7y9NhsVA==
-----END CERTIFICATE-----
```
Robert’s private key:
--------BEGIN RSA PRIVATE KEY--------
MIIEowIBAAKCAQEA09wUaWtxCSwLD25VCMFkIGbn5wBja56xQW2hnMQU2xz/hT
W6UNduyXuhYq7n+1XfttS5+P0j30F0ac+iMTk7zXR5EgaBj0ju0DwM+oADWwyn
V44hkqN6LSFESnBuTc3Nb30bPvSyYPP9S/ZY+r+FAcycxX0FOE26ZpWvvdPCgtK
20Q6C6ltQziyYIEWefgEOncOpA7cfcsfj3aKtqw2/2n07lp/S72bRiWhh+JmO5Q0
Sxe+iFhJdjkw3NDq82Zyw29s5ah5C91b69Frk0gRzPGg2cd/MBZ9xQCeo152FhN7/
pq1Agd4z0qZruyu2UYhvwAzyWdcVjqTCML715WiDaQAAB0iBAAv+Q3GMUYPaRhbj
1tH+Ekw86MCuB2n879r9jbeCj3QJOA/CgkAGPkIF7zbFwnY8RTXjHgEAUWH+W+b
4PphGwynuUjfgP8RavfMVvYNS1dmsrBYWtd0a44mwDnBf7vec99U71X5j2Hh
r8NP7et8a0o0xFeY9G46WDkC0nKh8agMMVymY/ Vu2KpH0f01hTPlmxS7W+e+d3Q
mva1MGue+EL079upgokhr4E0036Ce4luCnqQfOAkscYMcK7Y5guce620IYLE
Cqee2PE8eqWwhSN1981CF15aEbo0ApMcMwrfcbpQMQHqUyQh2XvqegW0qGQLn
UA0i6NEcgyE9ATfrFg3Kwu1fK+KxtX61Mj0W7yGn434nTb/9+xsKcoc0i6LoPbO
VHSVqHPjijicbCuyuA77K616Ahv7AV0s2FRHAb3M7wOYVgtT52+12a4F6HEM4U42
ISAcsS4vCfhYq10c91bY1XkXxErp0yBlReEaSAL6NhArAEgWccQgYaE3ASod
gEcahQEnu5P8Uy59jy9AfBArxVdqKwhM02trKflkYgvtv7nE3i3EMogJvHg23nr5iS
IpwxGqBiQveEGuVo3D0Jc5cZET0wewWbIec/CtZfnnbCnCNx8jwX5/cmTzMUHvUs
VJ1wpUdn+k7+G8hK10K+RpQdOCxXptHRlKQPBeCucYAYvCu1FLB83BvdQfUpKlo
TZEpak5bdyj7Z1F1PzunJyygP+tonr78TtfAliPogjr5g7lWnsL8BNTEzvRqSr
iuqW39EzXVPnHvUXaazs3RPNob1RaJwIrjx0046m4137xWeUJe/9I9C59QLoSjWJn
2f+ntWpmp8gDrF/F6jfL+LQkbgqcYdafi2Ef/cHcMiuHexUHr5s4kcktGofe57RdI
hQNdYP2Nv9usnrtvyn2J5j6JGgksF+QtvT3lqygShk4Vt43H1TnB/A5p5bb
//Mu2ezQru9a5UMkE0LjDnbCBE/kFhH62WUp7F5k/C/ FHYg9Q3u0v7PCAYy
CuFNQKQkgBw2k5sdVn4iwNv4xwEllhia+141zg8wpjJ1UxncOv1dGzA2nCnW9
wpOr+jvkh6Kv6X1im0tcqctZ07pC3JbEAtAcHhji21k+ZAEjQMFe/h62Rcv6sbozq0
5dFcb2wZie21Qomq3j8+0yILss/uzFkGjJloJrP+OtPKSrf+r/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 --
H4sIAIPaU0E0CA+ybeUTaXz7Hc5CIHpoqSIqVFCu5tsDhAEDATQhCsQEXZ
JBTiyGUSIREURU8i1LZQkyYERVHucKlUWPlvusXjCeeI3lfCauRaUPSFf8
tjXH/JPmdm3fTjy8/n8f8J6TEK5VCCYqTKCMd+YhKp/O1ABEAghb8Eki
wp89hNsiQAXcIaHdBACGIRDAc1CBqC7yAGdV6HEKFWIqsVrKnKISD9z8XvT
jd8F+H+Cyze10r+Bgo5oXvIm0U0fHSITRMnRjuyktRtFqQWb4btPa7yNoj
Vig4/37mxBwTgaUip2NHyBp7mEAAFF24CkTKI3LVLJK0BO5YHj9MgkkaHui
CAXASvAc3kwwGQdQbzu8YXhyVymULOCAIimgQFaDQeUo08Yo0RMF0M0NJ2hqm
zBk7qnu+zun28FbBJL+1C8XQAhAH8h3aeToLOmkoi1XKXWASAgEHImPCyYjho
l+qMYUije49g3dwp/ky9mM33V2mOAwWTdCpQPQSoSupspp0yjbZ1raj9Zg

Jennings, et al.  Informational  [Page 64]
RFC 6216              SIP Secure Call Flows               April 2011

dLIzlkwg4bG/vSfThh48HipXOWM1MKWVUI4oVE5KYaQUSTgVah45QXI0Kp
AxaRY8VChIq5yqRO8miekOzrmGf1iv5ccomZhCxlVl1Ut2qFbJMQjkK1apw
zh4g5+sduXbLMvI+ooh3JQkJaAI0LUahRaNWiM1NBFoUrUjnVRi9g11wU81
MrHrWd29QpEeQooRoAM/fmwcKo10URWffsPbN+2Bzwz33rxeH79kx4btOFAK
UXASQ6g6T8YXS8GcsVY99rXbpa8qFwsvs/opz/TRYIPUdWdFY1H0U5n8U6q
UCmXXYN92j9eQqNFQ6LQ1rxe/KejQrQ8s8hYrnpncrcoxoQ1/AqEYbc3KZ7A8k1w5
b2A0sUsn5YupxI3A02s5Fw1xxeDAY0A4Q7Nu1830dPsdxh8wvYDjmCztdJdr
LwGmAnMlGD4Ay0uRN7DqQGkDQdZphPvuJr80uAfAppAKEdAc6N1qa2JSPR1R
Yv1kShQA0Awagkx7mHTJniiCrQHQhJvUG7tjgcxxosUxugKQOQg0aDhtafY
jxv1EWhGmUXMc52+0ZbyYES/T9QMPFCruunoJ2NOVbV7yowrVxOAssZ
RsQC4fra+i+2b6bundvTtrfKzY2QnqRCseulYuGo/vk/OP3EcLL1nVrE5
5wu+o+xwvmm/2bJi7N5U50VKuelpKp/cGMCQ9cT11U6qWM9q8sg6i/o3247
9cw/y/DPu53Il8dEFMXNfdLEavBldeOe/9kpe41EmjfE19+Wu45kWR6brjci
VQd9+x6B01nlp9f2h+7Mu2VcCtCq+Wg8137En2mVFXX0RsCsmiSSCuPqja+J
5dXw7+/51izHRzbrtr7+66y1i/wN1rp2q4e0c4P7AeX0FRtHFxeEgoPoYt
+2FPliktnkFy6q2j90ssmpH+vgv9y9jo77iXooaaVBic0292t36EWCkPVJIPt
izU4aBq3uQd4EaQ1P/AEEOENM/DAH41v7/HeU3PgmBhZ/9x3bJn/lgP15hokM
IwChJFAsQmgeUsli7xH/SVh1C1h3gihE2MhugA+QYBqEmoJe2QlPq3VeA+i
UPAPEM/F/KjdCp3p/yj1vveyh1YwFw7eBrWU10j0wKwKB6QmTSEMkAKH
GflB190tS1+8bw8H9HYeZe2Fw/0g9/2/0fEf9Ed+HhrvN/+N/vmdka15A
B6owxNxbr8x8//g58U14BC1I79+XKG7Tqd3G7y/VodEx0FHgRht/e8/fJgJd
PR79pR+8PcroHddFfKhmMcASABdCw2Itr+t/gg/59QPEKDOaataTHGY1v/it/F0CJE
FPFvjAgxjPEGMWA8ncUmMCSq0qZI0WRniUimmU0mX84f8L8ESUC8EmScDI1BCR
O1L/+f3Rr/eit/t2W6y/9vf/8/&Hfp/vNPV5evaR6TYR7VRFWlfhu119PHE17
3jz7Is3BmbkObUg3w2qptwGq9cbbC6Oh96zemFj4aapsSHbkVgcoUcethTFK
OWy7eC0CPaU/puxGqxdmg9Z2P0d0LbDlbNFv/jv/vW+nunt/romayapSsrV8D8yrk
neeNqSNr8a8dKxkop2bOrs3FLyDEzeaIqUusqMYF5P524dFbywytua2/eLE
HIKUnbddeqamG+9fWn8y8KrFgO2Dhjr1PcK2W35634htWhFR3J3aEn7x7eb2
u4b1leyw5s1ilzZ/Pt/LAI4khZhN2eNd92Y9k604RuOKwaQ7t0VP5UOG7KnAvb
MxPU05e9hVybmpdFite7tcit/u/USkuDv/A/2/kfyp7US1PESx9fef2YBgBr/C
K5rnx6lIt6uJ+A/Sa4DFD3RJPB88pEIEFehp9wzh/Af5EB8gdx77v8r8qX5r
AIM2Zd11qCbbFaLVEgg+YXXwCWFxm2s6Adb5d63F0pIA+B88PEY7F/1v/A9L+
FNHa0m6DL2G1U1GvcV1caypkc5e00SWyTKYklf1Q5SBMfupKyU+BGJepJ
uqG01Vta6G0B+H0vddh6Uhe/hp0zOSju10tZe9XrmM7+mnrnDUfqaajutiR+
1V3a4n2nJLBOopecHmxeMn38k05v31pmFmpbap222DaH10dSCJMOD
vtxvaEGU1R33svcU2d951Y2cOPDjpzpej526zb1xcmkkcJduqTvhzi77
bUFR48K92gsN+Jcs1aBzgpe997xylet+ml1+dHxyR0EGvRS800f145vt
Tw3N/Q5mpgyp9KdU067/Vmnh5qGICyaIApp8WjVyscdj49ps/sk/eTD49W
xx2U4u05U1wO0X07T3xbD5110g3fnw70K1DdwZb1D031LNd5d05yrb/d6b
q+y8dkcwoTfGNYxvum/Oj1qIMgwaWNJK3PSPBHe1tG6F93Xb9x59K6M6d
a4o5rgrXG71KCoistL1eM1ITYMEyrF/XQKKK/0vEhIy2nwiTIFQ2uU+hDpeqec
vb1j9hr+Zd25Quo82ZJf111vnQ+v7swelWN7g0irbvoHyN18mp7v/PGI
rJUDr/7/Lv531A0Atv1/uln/d/DBLb5L3A5v5r0feAI8JPAPUCQA0ARX+4zr4E
/xSRO614he9/7fzaBzvJ/V7NC0P5M5AhioDl0hC8RO80YC0+eAbkPwEwFgkF/
J+18P/xRATSyis/Hw8QBGQSY7+P6HR/29uVr39/72M/3v9/3f/8f8j2qD
Df5/g3R9K6767TPcAKzXPt5r4gA0Nj1yVwagPrhA84HzyWMTfnnMM+DwK0
pxBoq/94Oev7V8It1vX3/87AfxkICwoJBI1k0A8BvX94PG7jFj+eG1JBEj1Xh
P3y3B1L1LzdYdGtw/w/mMD1CjihXkKd6sDsLzWHzHqm1J8s+X+t/7/2ACqgJv

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