Example call flows using SIP security mechanisms
draft-jennings-sip-sec-flows-00

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

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

This Internet-Draft will expire on April 18, 2004.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document shows call flows demonstrating the use of SIPS, TLS, and S/MIME in SIP. This draft provides information that helps implementors build interoperable SIP software. It is purely informational. To help facilitate interoperability testing, it includes certificates used in the example call flows and a CA certificate to create certificates for testing.

Warning - this is a very early draft of this document. The call flows in it have not been verified against multiple versions of the software and have reasonable odds of being wrong.
Table of Contents

1.  Conventions ................................................. 3
2.  Introduction ............................................. 3
3.  Security Considerations ................................. 3
4.  CA Certificates ........................................... 4
5.  Host Certificate .......................................... 7
6.  Callflow with Message over TLS ......................... 8
7.  Callflow with TLS with Mutual Authentication .......... 10
8.  User Certificates ......................................... 10
9.  Callflow with Signed Message ............................ 14
10. Callflow with Encrypted Message ......................... 19
11. Callflow with Signed and Encrypted Message .......... 20
12. Callflow with SRTP keying material in the SDP ....... 26
13. Callflow with secure REFER ............................... 26
14. Test Notes ................................................ 26
15. Making Test Certificates ................................. 27
16. makeCA script ............................................. 28
17. makeCert script ........................................... 30
18. Certificates for testing .................................. 32
19. Message Dumps ............................................ 35
20. Open Issues ............................................... 39
21. Still To Do ................................................ 39
22. Acknowledgments ......................................... 40

Normative References ........................................... 40
Informative References ........................................ 40
Author’s Address .............................................. 41
Intellectual Property and Copyright Statements ........... 42
1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

2. Introduction

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

A simple SIP call flow using SIPS and TLS is shown in Section 6. The certificates for the hosts used are shown in Section 5 and the CA certificates used to sign these are shown in Section 4.

The text from Section 9 through Section 11 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 8 and are signed with the same CA certs.

A way to make certificates that can be used for interoperability testing is presented in Section 15, along with methods for converting these to various formats.

In Section 14, a partial list of things implementors should check that they do in order to implement a secure system is presented.

Binary copies of various messages in this draft that can be used for testing appear in Section 19.

3. Security Considerations

Implementators 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 implementors 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 top mistakes that have been seen at the SIPit interoperability events. Just because an implementation does everything this document recommends does not make it secure.

4. CA Certificates

The certificate used by the CA to sign the other certificates is shown below. This is a X509v3 certificate. Note that the basic constraints allow it to be used as a CA.

Version: 3 (0x2)
Serial Number: 0 (0x0)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
Ou=Sipit Test Certificate Authority
Validity
  Not Before: Jul 18 12:21:52 2003 GMT
  Not After : Jul 15 12:21:52 2013 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
Ou=Sipit Test Certificate Authority
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (1024 bit)
    Modulus (1024 bit):
      00:c3:22:1e:83:91:c5:03:2c:3c:8a:f4:11:14:c6:
  Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Key Identifier:
  X509v3 Authority Key Identifier:
  DirName:/C=US/ST=California/L=San Jose/O=sipit/
  OUsipit Test Certificate Authority
serial:00
X509v3 Basic Constraints:
CA:TRUE
Signature Algorithm: sha1WithRSAEncryption

6f:52

The ASN.1 parse of the CA certificate is shown below.

0:l= 804 cons: SEQUENCE
4:l= 653 cons: SEQUENCE
8:l=  3 cons:  cont [ 0 ]
10:l=  1 prim: INTEGER           :02
13:l=  1 prim: INTEGER           :00
16:l= 13 cons: SEQUENCE
18:l=  9 prim: OBJECT            :sha1WithRSAEncryption
29:l=  0 prim: NULL
31:l= 112 cons: SEQUENCE
33:l=  11 cons:  SET
35:l=  9 cons:  SEQUENCE
37:l=  3 prim: OBJECT            :countryName
42:l=  2 prim: PRINTABLESTRING    :US
46:l=  19 cons:  SET
48:l=  17 cons: SEQUENCE
50:l=  3 prim: OBJECT            :stateOrProvinceName
55:l= 10 prim: PRINTABLESTRING    :California
67:l=  17 cons:  SET
69:l=  15 cons: SEQUENCE
71:l=  3 prim: OBJECT            :localityName
76:l=  8 prim: PRINTABLESTRING    :San Jose
86:l=  14 cons:  SET
88:l=  12 cons: SEQUENCE
90:l=  3 prim: OBJECT            :organizationName
95:l=  5 prim: PRINTABLESTRING    :sipit
102:l= 41 cons:  SET
104:l= 39 cons: SEQUENCE
106:l=  3 prim: OBJECT            :organizationalUnitName
111:l= 32 prim: PRINTABLESTRING    :
Sipit Test Certificate Authority
145:l= 30 cons: SEQUENCE
147:l=  13 prim: UTCTIME :030718122152Z
162:l=  13 prim: UTCTIME :130715122152Z
177:l= 112 cons: SEQUENCE
179:l=  11 cons: SET
181:l=  9 cons: SEQUENCE
183:l=  3 prim: OBJECT :countryName
188:l=  2 prim: PRINTABLESTRING :US
192:l=  19 cons: SET
194:l=  17 cons: SEQUENCE
196:l=  3 prim: OBJECT :stateOrProvinceName
201:l=  10 prim: PRINTABLESTRING :California
213:l=  17 cons: SET
215:l=  15 cons: SEQUENCE
217:l=  3 prim: OBJECT :localityName
222:l=  8 prim: PRINTABLESTRING :San Jose
232:l=  14 cons: SET
234:l=  12 cons: SEQUENCE
236:l=  3 prim: OBJECT :organizationName
241:l=  5 prim: PRINTABLESTRING :sipit
248:l=  41 cons: SET
250:l=  39 cons: SEQUENCE
252:l=  3 prim: OBJECT :organizationalUnitName
257:l=  32 prim: PRINTABLESTRING :
Sipit Test Certificate Authority
291:l= 159 cons: SEQUENCE
294:l= 13 cons: SEQUENCE
296:l=  9 prim: OBJECT :rsaEncryption
307:l=  0 prim: NULL
309:l= 141 prim: BIT STRING
00 30 81 89 02 81 81 00-c3 22 1e 83 91 c5 03 2c .0.......".....,
3c 8a f4 11 14 c6 4b 9d-fa 72 78 c6 b0 95 18 a7 <.....K..rx..
..y.]...!--------
5b 29 b3 90 13 73 66 92-6e df 4c b3 b3 1c 1f 2a []...sf.n.L....*
82 0a ba 07 4d 52 cd df-87 1a bd 71 eb e1 99 6a ....*v.....q...j
f9 2e 03 ff 2a 76 cd df-87 1a bd 71 eb e1 99 6a ....*v.....q...j
c4 7f 8e 74 a0 77 85 04-e9 41 ad fc 03 b6 17 75 ...t.w...A.....
ua 33 ea 0a 16 d9 fb 79-32 2e f8 cf 4d c6 34 a3 .3.....y2...M.4.
ff 1b d0 68 28 e1 9d e5-02 03 01 00 01 ...h(........
453:l= 205 cons: cont [ 3 ]
456:l= 202 cons: SEQUENCE
459:l=  29 cons: SEQUENCE
461:l=  3 prim: OBJECT :X509v3 Subject Key Identifier
466:l=  22 prim: OCTET STRING
04 14 6b 46 17 14 ea 94-76 25 80 54 6e 13 54 da ..kF....v%.Tn.T.
a1 e3 54 14 a1 b6 ....T....
490:l= 154 cons: SEQUENCE
493:l=  3 prim: OBJECT :X509v3 Authority Key Identifier
498:l= 146 prim: OCTET STRING
5. Host Certificate

The certificate for the host b.example.com is shown below. Note that the Subject Alternative Name is set to b.example.com and is a DNS type.

Data:

Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit, OU=Sipit Test Certificate Authority
Validity
  Not Before: Jul 20 20:46:16 2003 GMT
  Not After : Jul 19 20:46:16 2004 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=b.example.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
fb:32:7b:00:b1:10:64:19:2a:ed:3e:d9:19:7f:bd:
Exponent: 65537 (0x10001)

6. Callflow with Message over TLS

The flow below shows the edited SSLDump output of the host a.example.com forming a TLS connection to b.example.com. In this example mutual authentication is not used. Note that the client proposed three protocol suites including the required TLS_RSA_WITH_AES_128_CBC_SHA. The certificate returned by the server contains a Subject Alternative Name that is set to b.example.com. A detailed discussion of TLS can be found in [9].
New TCP connection #1: a.example.com(5071) <-> b.example.com(5081)

1 1  0.0015 (0.0015)  C>SV3.1(49)  Handshake
   ClientHello
      Version 3.1
      random[32]=
         3f 1d 41 76 31 6f af f1 42 fa 7b 57 c7 79 49 2b
d4 21 9c be e9 8b 85 83 56 4b 36 cb f2 99 ef b2
      cipher suites
         TLS_RSA_WITH_AES_256_CBC_SHA
         TLS_RSA_WITH_AES_128_CBC_SHA
         TLS_RSA_WITH_3DES_EDE_CBC_SHA
      compression methods
         NULL

1 2  0.4307 (0.4292)  S>CV3.1(74)  Handshake
   ServerHello
      Version 3.1
      random[32]=
         3f 1d 41 77 92 f5 55 a3 97 5f bash 0a 3c 00
         bc 0c 59 91 1c 6b 2b 4a 0e 98 40 21 a9 b5 4b 6f
      session_id[32]=
         10 3c 8c aa 75 d8 62 0b c3 5b ad 24 c1 7f 4f 80
         25 b7 1c 40 a3 3c e1 85 0d b5 29 d3 15 40 51 d3
      cipherSuite         TLS_RSA_WITH_AES_256_CBC_SHA
      compressionMethod                   NULL

1 3  0.4307 (0.0000)  S>CV3.1(822)  Handshake
   Certificate
      Subject
         C=US
         ST=California
         L=San Jose
         O=sipit
         CN=b.example.com
      Issuer
         C=US
         ST=California
         L=San Jose
         O=sipit
         OU=Sipit Test Certificate Authority
      Serial         01
      Extensions
         Extension: X509v3 Subject Alternative Name
         Extension: X509v3 Basic Constraints
         Extension: X509v3 Subject Key Identifier
         Extension: X509v3 Authority Key Identifier

1 4  0.4307 (0.0000)  S>CV3.1(4)  Handshake
   ServerHelloDone

1 5  0.4594 (0.0286)  C>SV3.1(134)  Handshake
   ClientKeyExchange
Once the TLS session is set up, the following MESSAGE message is sent from a.example.com to b.example.com. Note that the URI has a SIPS URL and that the VIA indicates that TLS was used.

MESSAGE sips:bob@b.example.com:5081 SIP/2.0
To: <sips:bob@b.example.com:5081>
From: <sip:alice@example.com>;tag=2639484b
Via: SIP/2.0/TLS b.example.com:5071;
     branch=z9hG4bK-c87542-240491824-1-c87542-
     Call-ID: 7ba3572175b0f542
CSeq: 1 MESSAGE
Contact: <sips:alice@a.example.com:5071>
Max-Forwards: 70
Content-Type: text/plain
User-Agent: SIPimp.org/0.2.1 (curses)
Content-Length: 2

Hi

The response is sent from b.example.com to a.example.com over the same TLS connections. It is shown below.

SIP/2.0 200 OK
To: <sips:bob@b.example.com:5081>;tag=514db9e7
From: <sip:alice@example.com>;tag=2639484b
Via: SIP/2.0/UDP b.example.com;
     branch=z9hG4bK-c87542-240491824-1-c87542-;received=127.0.0.1
     Call-ID: 7ba3572175b0f542
CSeq: 1 MESSAGE
Contact: <sips:bob@b.example.com:5081>
Content-Length: 0

7. Callflow with TLS with Mutual Authentication

8. User Certificates

Alice’s certificate is shown below. Note that it has a Subject Alternative Name of type email and is set to alice@a.example.com. In this example a.example.com is the domain for Alice, the message could
be coming from a host called host1.a.example.com, and the AOR in the user certificate would still be the same.

Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=US, ST=California, L=San Jose, O=sipit,
  OU=Sipit Test Certificate Authority
Validity
  Not Before: Jul 20 14:29:54 2003 GMT
  Not After : Jul 19 14:29:54 2004 GMT
  Subject: C=US, ST=California, L=San Jose, O=sipit,
  CN=alice@a.example.com
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (1024 bit)
    Modulus (1024 bit):
      e3:69:3b:f0:aa:1b:ad:f2:ab:1e:44:46:b2:8a:ab:
      ed:79:e5:81:19:c7:02:e2:1b
    Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Alternative Name:
    email: alice@a.example.com
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:
  DirName:/C=US/ST=California/L=San Jose/O=sipit/
  OU=Sipit Test Certificate Authority
  serial:00
Signature Algorithm: sha1WithRSAEncryption
Alice’s private key is shown below.

```
0: 604 cons: SEQUENCE
  4:  1 prim: INTEGER :00
  7: 129 prim: INTEGER :
      F09F919A6D6F81B9D67DB5FE953A298ACC733DBB97A33C8F952DD9913042BF19B
      C2F59372A9BE197FCC2D296D076DBB50E7B15974595BB073ADC864BD51C671A
      822FC2CF387D32B5ADCE63C8C27A0A86E7F4D86DD2B9BE393FBF2B1AEB
      4466BAB852CC811303980665570CFF3C4F02CBED79E58119C7002E1B
  139:  3 prim: INTEGER :010001
  144: 128 prim: INTEGER :
      4764C0F9D5E090D7F6E91ACOE4B638249D471E55BA3394EBDB7607C3E44D87904F
      4BE03B586B229722D6E52CC795A0BE7D90F81A99D51B248BF79DF8C655E48B135
      6249D82F9B18C37525FA05D3562399E4912BC902FA92CF12D7AE653C3COD851A4B
      B3DF35E872206460FC076E02D01D3CF23D1934100FEC7EAC72D89B
  275:  65 prim: INTEGER :
      FA5A76D62011E3A219B4D9CF2A392FF57A55BC4E1092EC67030E31ABEDC591485
      C284250BC0195C33A92920B340B2636EBB880C3DC62748A6045A07FCC2E97
  342:  65 prim: INTEGER :
      F60CEC61DB985C1AE0F927E831AADA2E12DF889D135E91A49B662B8094CF140075A
      9C782DF6A28F538D2C51CC4910CB02B159894FB597D17A3FB69DD37099D1D
  409:  64 prim: INTEGER :
      53E735A495A2E9334E823986801B2A0CC186FDB681E4DDF44B6D56EF83BFBD6B0F
      591DB887E3A9902A042B07622DCA64E5A33424701FCAB2A251B084A3ED89
  475:  65 prim: INTEGER :
      CB08F91E39E88A65C2D103AF6AB2E07771D2A5101F115A5E6C446D64873278719F
      4872B8E1A4D4C94C74B70AC3815792DA598754965764F69E9C9F03460EAA1
  542:  64 prim: INTEGER :
      021FC88DE2C3F4B82BE937CD45B819AE8C5777BFF14C74F719FBB83E567A563A
      9B2256EC3563E764B269D3C34BFEC772EB443484D974B8FF07C52D9BF95DC24
```

Bob’s certificate is shown below. Bob is in the domain b.example.com.

```
Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=US, ST=California, L=San Jose, O=sipit, OU=Sipit Test Certificate Authority
  Validity
    Not Before: Jul 20 14:30:06 2003 GMT
    Not After: Jul 19 14:30:06 2004 GMT
```
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=bob@b.example.com
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (1024 bit)
    Modulus (1024 bit):
      00:b0:ef:02:43:fd:59:28:0b:d3:59:ff:e6:66:3a:
      a7:30:b0:e5:11:54:c0:d7:e9:8a:51:a7:2b:30:94:
      b4:5e:b0:8a:el:ab:0d:c4:67:9b:2f:10:b1:c8:71:
      28:0b:0d:36:75:46:30:f9:17:39:d0:c8:e2:14:ac:
      0c:ce:fb:be:92:6b:d8:03:c3:e6:fb:25:78:ea:5c:
      18:76:36:06:ba:2e:78:cf:3d
    Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Alternative Name:
    email:bob@b.example.com
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Subject Key Identifier:
X509v3 Authority Key Identifier:
  DirName:/C=US/ST=California/L=San Jose/O=sipit/
  OU=Sipit Test Certificate Authority
  serial:00
Signature Algorithm: sha1WithRSAEncryption
  b4:ab
Bob’s private key is shown below.

0: 605 cons: SEQUENCE
  4:  1 prim: INTEGER    :00
  7: 129 prim: INTEGER    :
    B0EF0243FD59280BD359FFEE6663AA730B0E51154C0D7E98A51A72B309498EFBBF9
    8A95A6CA5EE37AAFA22AF9B45EB08AE1AB0DC4679B2F10B1C871280B0D36754630
    F917390D0C8E214ACECBBBA3DD1A75D13833ED375678EF36A55DB32371291594E8
    503CF87BA70CCCEF0BE926BD803C3E6FB2578E5A5C18763606BA2E78CF3D
9. Callflow with Signed Message

Example Signed Message. The value on the Content-Type line has been broken across lines to fit on the page but it should not be.

```
MESSAGE sip:bob@b.example.com SIP/2.0
To: <sip:bob@b.example.com>
From: <sip:alice@a.example.com>;tag=1b2f5769
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-730075406-1--c87542--;rport
Call-ID: 22b4f26debe23a0e
CSeq: 1 MESSAGE
Contact: <sip:alice@a.example.com:5070>
Max-Forwards: 70
Content-Type: multipart/signed;boundary=65b6563f5e8ef632;
        micalg=sha1;protocol=application/pkcs7-signature
User-Agent: SIPimp.org/0.2.2 (curses)
Content-Length: 1653

--65b6563f5e8ef632
Content-Type: text/plain
Content-Transfer-Encoding: binary

Hi
```
It is important to note that the data the signature is computed across includes the header and is:

```
Hi
```

The response follows. The Via line has been split across lines for formatting but it should not be.

```
SIP/2.0 200 OK
To: <sip:bob@example.com>;tag=6b167ed8
From: <sip:alice@example.com>;tag=1b2f5769
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-730075406-1--c87542--;rport=5070;received=127.0.0.1
Call-ID: 22b4f26d6be23a0e
CSeq: 1 MESSAGE
Contact: <sip:bob@example.com:5060>
Content-Length: 0
```

ASN.1 parse of binary blob 1. Note that at address 30, the hash for the signature is specified as sha1.

```
0: SEQUENCE
  4: OBJECT :pkcs7-signedData
  15: cont [ 0 ]
  19: SEQUENCE
  23: INTEGER :01
  26: SET
  28: SEQUENCE
  30: OBJECT :sha1
  37: NULL
  39: SEQUENCE
```
41: OBJECT :pkcs7-data
52: cont [ 0 ]
56: SEQUENCE
60: SEQUENCE
64: cont [ 0 ]
66: INTEGER :02
69: INTEGER :55018102490073
78: SEQUENCE
80: OBJECT :sha1WithRSAEncryption
91: NULL
93: SEQUENCE
95: SET
97: SEQUENCE
99: OBJECT :countryName
104: PRINTABLESTRING :US
108: SET
110: SEQUENCE
112: OBJECT :stateOrProvinceName
117: PRINTABLESTRING :California
129: SET
131: SEQUENCE
133: OBJECT :localityName
138: PRINTABLESTRING :San Jose
148: SET
150: SEQUENCE
152: OBJECT :organizationName
157: PRINTABLESTRING :sipit
164: SET
166: SEQUENCE
168: OBJECT :organizationalUnitName
173: PRINTABLESTRING :Sipit Test Certificate Authority
207: SEQUENCE
209: UTCTIME :031014202459Z
224: UTCTIME :061013202459Z
239: SEQUENCE
241: SET
243: SEQUENCE
245: OBJECT :countryName
250: PRINTABLESTRING :US
254: SET
256: SEQUENCE
258: OBJECT :stateOrProvinceName
263: PRINTABLESTRING :California
275: SET
277: SEQUENCE
279: OBJECT :localityName
284: PRINTABLESTRING :San Jose
294: SET
SEQUENCE
  OBJECT :organizationName
  PRINTABLESTRING :sipit

SET
  SEQUENCE
    OBJECT :commonName
    T61STRING :alice@a.example.com

SEQUENCE
  OBJECT :rsaEncryption
  NULL

BIT STRING con [ 3 ]

SEQUENCE
  OBJECT :X509v3 Subject Alternative Name
  OCTET STRING

SEQUENCE
  OBJECT :X509v3 Basic Constraints
  OCTET STRING

SEQUENCE
  OBJECT :X509v3 Subject Key Identifier
  OCTET STRING

SEQUENCE
  OBJECT :X509v3 Authority Key Identifier
  OCTET STRING

SEQUENCE
  OBJECT :sha1WithRSAEncryption
  NULL

BIT STRING

SET
  SEQUENCE
    INTEGER :01

SEQUENCE

SET
  SEQUENCE
    OBJECT :countryName
    PRINTABLESTRING :US

SET
  SEQUENCE
    OBJECT :stateOrProvinceName
    PRINTABLESTRING :California

SET
  SEQUENCE
    OBJECT : localityName
    PRINTABLESTRING : San Jose

SET
956: SEQUENCE
958:  OBJECT :organizationName
963: PRINTABLESTRING :sipit
970: SET
972: SEQUENCE
974:  OBJECT :organizationalUnitName
979: PRINTABLESTRING :Sipit Test Certificate Authority
1013: INTEGER :55018102490073
1022: SEQUENCE
1024:  OBJECT :sha1
1031: NULL
1033: cont [ 0 ]
1036: SEQUENCE
1038:  OBJECT :contentType
1049: SET
1051:  OBJECT :pkcs7-data
1064: SEQUENCE
1066:  OBJECT :signingTime
1075: SET
1077: UTCTIME :031015000907Z
1092: SEQUENCE
1094:  OBJECT :messageDigest
1105: SET
1107: OCTET STRING
1129: SEQUENCE
1131:  OBJECT :S/MIME Capabilities
1142: SET
1144: SEQUENCE
1146:  OBJECT :des-ede3-cbc
1158: SEQUENCE
1160:  OBJECT :rc2-cbc
1170: INTEGER :80
1174: SEQUENCE
1176:  OBJECT :rc2-cbc
1186: INTEGER :40
1189: SEQUENCE
1191:  OBJECT :des-cbc
1198: SEQUENCE
1200:  OBJECT :rc2-cbc
1210: INTEGER :28
1213: SEQUENCE
1215:  OBJECT :rsaEncryption
1226: NULL
1228: OCTET STRING
10. Callflow with Encrypted Message

Example encrypted message:

```
MESSAGE sip:bob@b.example.com SIP/2.0
To: <sip:bob@b.example.com>
From: <sip:alice@a.example.com>;tag=4bbaf0d
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-558422834-1--c87542--;rport
Call-ID: 132bb895019d4536
CSeq: 1 MESSAGE
Contact: <sip:alice@a.example.com:5070>
Max-Forwards: 70
Content-Disposition: attachment; handling=required; filename=smime.p7
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name=smime.p7m
User-Agent: SIPimp.org/0.2.2 (curses)
Content-Length: 385

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

The Response. The Via is split across lines for formatting but is not split in the real message.

```
SIP/2.0 200 OK
To: <sip:bob@b.example.com>;tag=330805f5
From: <sip:alice@a.example.com>;tag=4bbaf0d
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-558422834-1--c87542--;\rport=5070;received=127.0.0.1
Call-ID: 132bb895019d4536
CSeq: 1 MESSAGE
Contact: <sip:bob@b.example.com:5060>
Content-Length: 0

ASN.1 parse of Binary Blob 2. Note that at address 323, the encryption is set to des-ebe3-cbc.

```
0: SEQUENCE
4: OBJECT :pkcs7-envelopedData
15: cont [ 0 ]
19: SEQUENCE
23: INTEGER :00
26: SET
```

Jennings          Expires April 18, 2004          [Page 19]
11. Callflow with Signed and Encrypted Message

Example Signed and Encrypted Message

In the example below, one of the headers is contained in a box and is split across two lines. This was only done to make it fit in the RFC format. This header should not have the box around it and should be on one line with no whitespace between the "mime;" and the "smime-type". Note that Content-Type is split across lines for formatting but is not in the real message.
MESSAGE sip:bob@b.example.com SIP/2.0
To: <sip:bob@b.example.com>
From: <sip:alice@a.example.com>;tag=1d8673a3
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-488884104-1--c87542--;rport
Call-ID: 450c8b112715a732
CSeq: 1 MESSAGE
Max-Forwards: 70
Content-Type: multipart/signed;boundary=75b3d73b4e24d3f6;\micalg=sha1;protocol=application/pkcs7-signature
User-Agent: SIPimp.org/0.2.2 (curses)
Content-Length: 2158

--75b3d73b4e24d3f6
<p>|---See note about stuff in this box---------------------|
|Content-Type: application/pkcs7-mime;|</p>
<table>
<thead>
<tr>
<th>smime-type=enveloped-data;name=smime.p7m</th>
</tr>
</thead>
</table>
Content-Disposition: attachment;handling=required;filename=smime.p7
Content-Transfer-Encoding: binary

****************************
* BINARY BLOB 3 *
****************************
--75b3d73b4e24d3f6
Content-Type: application/pkcs7-signature;name=smime.p7s
Content-Disposition: attachment;handling=required;filename=smime.p7s
Content-Transfer-Encoding: binary

****************************
* BINARY BLOB 4 *
****************************
--75b3d73b4e24d3f6--

Response back. Note that the Via is split across lines for formatting.

SIP/2.0 200 OK
To: <sip:bob@b.example.com>;tag=40d7131b
From: <sip:alice@a.example.com>;tag=1d8673a3
Via: SIP/2.0/UDP 127.0.0.1:5070;branch=z9hG4bK-c87542-488884104-1--c87542--;rport=5070;received=127.0.0.1
Call-ID: 450c8b112715a732
CSeq: 1 MESSAGE
Contact: <sip:bob@b.example.com:5060>
Binary blob 3

0: SEQUENCE
  4: OBJECT :pkcs7-envelopedData
15:  cont [ 0 ]
19:  SEQUENCE
23:  INTEGER :00
26:  SET
30:  SEQUENCE
34:  INTEGER :00
37:  SEQUENCE
39:  SEQUENCE
41:  SET
43:   SEQUENCE
45:    OBJECT :countryName
50:     PRINTABLESTRING :US
54:    SET
56:   SEQUENCE
58:    OBJECT :stateOrProvinceName
63:     PRINTABLESTRING :California
75:   SET
77:   SEQUENCE
79:    OBJECT :localityName
84:     PRINTABLESTRING :San Jose
94:   SET
96:   SEQUENCE
98:    OBJECT :organizationName
103:     PRINTABLESTRING :sipit
110:   SET
112:   SEQUENCE
114:    OBJECT :organizationalUnitName
119:     PRINTABLESTRING :Sipit Test Certificate Authority
153:    INTEGER :55018102490072
162:   SEQUENCE
164:    OBJECT :rsaEncryption
175:   NULL
177:   OCTET STRING
308:   SEQUENCE
310:    OBJECT :pkcs7-data
321:   SEQUENCE
323:    OBJECT :des-ede3-cbc
333:   OCTET STRING
343:    cont [ 0 ]
Binary Blob 4

0: SEQUENCE
4:  OBJECT :pkcs7-signedData
15:  cont [ 0 ]
19:  SEQUENCE
23:   INTEGER :01
26:  SET
28:  SEQUENCE
30:   OBJECT :sha1
37:   NULL
39:  SEQUENCE
41:   OBJECT :pkcs7-data
52:  cont [ 0 ]
56:  SEQUENCE
60:  SEQUENCE
64:   cont [ 0 ]
66:    INTEGER :02
69:    INTEGER :55018102490073
78:  SEQUENCE
80:   OBJECT :sha1WithRSAEncryption
91:   NULL
93:  SEQUENCE
95:  SET
97:   SEQUENCE
99:    OBJECT :countryName
104:   PRINTABLESTRING :US
108:  SET
110:   SEQUENCE :stateOrProvinceName
112:   OBJECT :printableString :California
129:  SET
131:   SEQUENCE
133:    OBJECT :localityName
138:   PRINTABLESTRING :San Jose
148:  SET
150:   SEQUENCE
152:    OBJECT :organizationName
157:   PRINTABLESTRING :sipit
164:  SET
166:   SEQUENCE
168:    OBJECT :organizationUnitName
173:   PRINTABLESTRING :Sipit Test Certificate Authority
207:  SEQUENCE
209:   UTCTIME :031014202459Z
224:   UTCTIME :061013202459Z
239:  SEQUENCE
SET
SEQUENCE
OBJECT :countryName
PRINTABLESTRING :US
SET
SEQUENCE
OBJECT :stateOrProvinceName
PRINTABLESTRING :California
SET
SEQUENCE
OBJECT :localityName
PRINTABLESTRING :San Jose
SET
SEQUENCE
OBJECT :organizationName
PRINTABLESTRING :sipit
SET
SEQUENCE
OBJECT :commonName
T61STRING :alice@a.example.com
SEQUENCE
OBJECT :rsaEncryption
NULL
BIT STRING
cont [ 3 ]
SEQUENCE
OBJECT :X509v3 Subject Alternative Name
OCTET STRING
SEQUENCE
OBJECT :X509v3 Basic Constraints
OCTET STRING
SEQUENCE
OBJECT :X509v3 Subject Key Identifier
OCTET STRING
SEQUENCE
OBJECT :X509v3 Authority Key Identifier
OCTET STRING
SEQUENCE
OBJECT :sha1WithRSAEncryption
NULL
BIT STRING
SEQUENCE
OBJECT :01
SEQUENCE
SEQUENCE
SEQUENCE
901: SET
903: SEQUENCE
905: OBJECT :countryName
910: PRINTABLESTRING :US
914: SET
916: SEQUENCE
918: OBJECT :stateOrProvinceName
923: PRINTABLESTRING :California
935: SET
937: SEQUENCE
939: OBJECT :localityName
944: PRINTABLESTRING :San Jose
954: SET
956: SEQUENCE
958: OBJECT :organizationName
963: PRINTABLESTRING :sipit
970: SET
972: SEQUENCE
974: OBJECT :organizationalUnitName
979: PRINTABLESTRING :Sipit Test Certificate Authority
1013: INTEGER :55018102490073
1022: SEQUENCE
1024: OBJECT :sha1
1031: NULL
1033: cont [ 0 ]
1036: SEQUENCE
1038: OBJECT :contentType
1049: SET
1051: OBJECT :pkcs7-data
1062: SEQUENCE
1064: OBJECT :signingTime
1075: SET
1077: UTCTIME :031015000922Z
1092: SEQUENCE
1094: OBJECT :messageDigest
1105: SET
1107: OCTET STRING
1129: SEQUENCE
1131: OBJECT :S/MIME Capabilities
1142: SET
1144: SEQUENCE
1146: OBJECT :des-ede3-cbc
1148: SEQUENCE
1150: OBJECT :rc2-cbc
1152: INTEGER :80
1174: SEQUENCE
1176: OBJECT :rc2-cbc
12. Callflow with SRTP keying material in the SDP

Still TODO.

13. Callflow with secure REFER

Still TODO.

14. Test Notes

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

A common problem in interoperability is that some SIP clients do not support TLS and only do SSLv3. Check that the client does use TLS.

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

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

Some implementations used binary MIME encodings while others used base64. There is no reason not to use binary – check that your implementation sends binary and preferably receives both.
15. Making Test Certificates

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

The instructions assume a Unix-like environment with openssl installed, but openssl does work in Windows too. Make sure you have openssl installed by trying to run "openssl". Run the makeCA script found in Section 16; 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 not ready to make certificates.

To create certs for use with TLS, run the makeCert script found in Section 17 with the fully qualified domain name of the proxy you are making the certificate for. For example, "makeCert host.example.net". This will generate a private key and a certificate. The private key will be left in a file named host.example.net_key.pem in pem format. The certificate will be in host.example.net_cert.pem. Some programs expect both the certificate and private key combined together in a PKCS12 format file. This is created by the script and left in a file named host.example.net.p12. Some programs expect this file to have a .pfx extension instead of .p12 - just rename the file if needed.

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 demoCA/cacert.pem and a PKCS#7 version of it is in demoCA/cacert.p7c.

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 with to PKCS#7 with "openssl crl2pkcs7 -nocrl -certfile cert.pem -certfile
IE, Outlook, and Netscape can import and export .p12 files and .p7c files. You can convert a pkcs7 certificate to PEM format with "openssl pkcs7 -in cert.p7c -inform DER -outform PEM -out cert.pem".

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

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

When validating a chain of certificates, make sure that the basic constraints on any non leaf node allow the certificate to be used for a CA. For example, if the domain example.com issues a certificate for alice@example.com, Alice should not be able to use this to sign a certificate for bob@example.com.

16. makeCA script

## Appendix B - makeCA script

```bash
#!/bin/sh
#set -x

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

CONF=${OPENSSLDIR:=/usr/local/ssl}/openssl.cnf
if [ ! -f $CONF ]; then
    echo "Can not find file $CONF - set your OPENSSLDIR variable"
    fi
    cp $CONF openssl.cnf
```
cat >> openssl.cnf <<EOF
[ cj_cert ]
subjectAltName=\${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
EOF
cat > demoCA/private/cakey.pem <<EOF
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,4B47A073ADE342E
aHm1Pa+ZrOV6v+JK05C1zxpxogG3j02uyoVf9rzq2bsZkVBKLU6xhwWjMDqAw8dH3fCrLhG Mixed
subjectAltName=\${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
EOF
cat > demoCA/cacert.pem <<EOF
-----BEGIN CERTIFICATE-----
MIIDJDCCAo2gAwIBAgIBADANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzETMBEGA1UE
subjectAltName=\${ENV::ALTNAME}
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
EOF
cat > demoCA/cacert.pem <<EOF
-----BEGIN CERTIFICATE-----
MIIDJDCCAo2gAwIBAgIBADANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzETMBEGA1UE

Jennings                 Expires April 18, 2004                [Page 29]
-----END CERTIFICATE-----
EOF

# uncomment the following lines to generate your own key pair

# openssl req -newkey rsa:1024 -passin pass:password \
#   -passout pass:password \
#   -sha1 -x509 -keyout demoCA/private/cakey.pem \
#   -out demoCA/cacert.pem -days 3650 <<EOF
#US
#California
#San Jose
#sipit
#Sipit Test Certificate Authority
#
#EOF
openssl crl2pkcs7 -nocrl -certfile demoCA/cacert.pem \ 
   -outform DER -out demoCA/cacert.p7c

17. makeCert script

Appendix C - makeCert script

#!/bin/sh
#set -x

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

ADDR=$1

echo "making cert for $ADDR"

rm -f ${ADDR}_*.pem
rm -f ${ADDR}.p12
case ${ADDR} in
  *:* Type="URI" ;;
  *@* Type="email" ;;
  *) Type="DNS" ;;
esac
rm -f demoCA/index.txt
touch demoCA/index.txt
rm -f demoCA/newcerts/*

#setenv ALTNAME "URI:${ADDR}"
#setenv ALTNAME "email:${ADDR}"
#setenv ALTNAME "DNS:${ADDR}"
ALTNAME="$TYPE:${ADDR}"
export ALTNAME

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

if [ $DAYS = 0 ]; then
  openssl ca -extensions cj_cert -config openssl.cnf \
    -passin:password -policy policy_anything \
    -md sha1 -batch -notext -out ${ADDR}_cert.pem \
    -startdate 990101000000Z \n    -enddate 000101000000Z \n    -infiles ${ADDR}_req.pem
else
  openssl ca -extensions cj_cert -config openssl.cnf \
    -passin:password -policy policy_anything \
    -md sha1 -days $DAYS -batch -notext \
    -out ${ADDR}_cert.pem \n    -infiles ${ADDR}_req.pem
fi
openssl pkcs12 -passin pass:password \
  -passout pass:password -export \
  -out ${ADDR}.p12 -in ${ADDR}_cert.pem \
  -inkey ${ADDR}_key.pem -name TheName \
  -certfile demoCA/cacert.pem

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

18. Certificates for testing

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

Alice’s certificate.

-----BEGIN CERTIFICATE-----
MIIDNDCCAp2gAwIBAgIBATANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaZVcm5pYTEMA8GA1UdDwEB/wQEAwIGA1UdIAwEBzAAMB0GA1UdDwEB/wQEAwIGA1UdIwEB/wQFAwQ lasts bytes removed for legibility

-----END CERTIFICATE-----

Alice’s private key

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgDq DwNzvBoRq5FEhE4Lw6mGZ9G1QeZjE5jXK06K5m63k8jJXmK8

-----END CERTIFICATE-----

Alice’s private key

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgDq DwNzvBoRq5FEhE4Lw6mGZ9G1QeZjE5jXK06K5m63k8jJXmK8

-----BEGIN CERTIFICATE-----
MIIDNDCCAp2gAwIBAgIBATANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaZVcm5pYTEMA8GA1UdDwEB/wQEAwIGA1UdIAwEBzAAMB0GA1UdDwEB/wQEAwIGA1UdIwEB/wQFAwQ lasts bytes removed for legibility

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDNDCCAp2gAwIBAgIBATANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaZVcm5pYTEMA8GA1UdDwEB/wQEAwIGA1UdIAwEBzAAMB0GA1UdDwEB/wQEAwIGA1UdIwEB/wQFAwQ lasts bytes removed for legibility

-----END CERTIFICATE-----

---

Internet-Draft              SIP Secure Flows                October 2003

Jennings                 Expires April 18, 2004                [Page 32]
Bob’s certificate

-----BEGIN CERTIFICATE-----
MIIDMDCCApmgAwIBAgIIBQ0xGzAUBgNVHQgCQjAvMRMwEQYDVQQIEwMTMDoGCSqG
SIb3DQEBAUAA4GCAwEBBQwFADoKMIIBIgIBAgIBATANBgkqhkiG9w0BAQUFAAOCAQ
-----END CERTIFICATE-----

-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCw7wJD/VkoC9NZ/+ZmOqcwsOURVMDx6YpRpyswljyv/mklabK
-----END RSA PRIVATE KEY-----
Certificate for a.example.com.

-----BEGIN CERTIFICATE-----
MIIDDCCCApGwIBAIAIBATANBgkqhkiG9w0BAQUFAwIwMQswCQYDVQQHEwJVUzEiTMBEGA1UECMQKQ2FsaWVucpMBMIHJQVQGEmwCwYDVQQDBQJUZXN0IENlcnRpZmljYXRlIEZhY3Rpb24gQ29udGVudDAwDgYDVQQLDwJHbG9nYXRpb24gQ29udGVudDQwLjAAMIIBIwIjANBgkqhkiG9w0BAQsFAAOCAQEEKeu4ZlVitOEFp5LSUf1PjUO7b0P1VBIDxxwH/DS4rYf8iGyR0nK5iN5TkL/2oOAVXz2h5H7Rf/93oC1xW0U6i7bCLsKzP+cvsYm2JZzcR78wWKTdA0i3U6kGdA9V/Hk5YI6/AuBQ6QxYlVz0+0hZ5Gq68ct9lMN03G8msuFM9P9CJh6C0+F/9JgR3gH/3AzOj3lM0fV7hUvMEkC0gUa9dQhGvW0Jm8wOj4AMeLx8M4P7BnBcP6Q1+bc50ww4HJqy5wMVbJjZc6Hj9h+J6uE04UoC6oMNk1OgoJfF+G79dIc1R0Y3n50Fpc6g3u57sMw0mggQF1FHwJz3aCq6f0iuVnW5Pw5H0JAc4iUa7f7qTeh4H0H742SxwHjxZB9za7fSTs0X1uhY+2Uz90zDP2Alr6/pc0fmmSe9U3d4K7jQf1uo80jav2QcSUF12fZdE0R6pStez795q+xwE8/8hZfAe/6vPmTkk+IFkGBv094zDowbe3JNvN5h9Ew2tj/3XQ0kM/Tgr0JrKJj scraped text here

-----END CERTIFICATE-----

Private key for a.example.com.

-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQDQXnzK3my3is16QFX81IXwdeTEWvGxSAiHwhtfpsa+bZk3LQ6yPyTdETTkpdKzy5+GxxKc+7nOAzb3BwdpruvQ+aI8M/SDNFyghChXwqgNBMs5g7B1dGP1tICc/26GEMU1enHk7enqnsP97Tqnr94eh5K1jAUKQ0x2nF0bUwIAQABAoGAltyBLlLVmnWxDh31i887DY/Auo7Me22WnHxHQCtSMIPtQsbj9Rjz3J6b4y19lK5SghDwDdUvfrZxx7KXL3zbrdHuM2pQB7eR0mLiiQPkF7zX4FDFpDfjekufq7jPJehl7ZYH5GYPAzRQXa/SAX/NOSKMhQLYiwaJ5bRECCQD6zRL/vPAhAjC7OD0yFstTbdFgVzT2lGhnaAh9FpMiCSr7Uy7VRRr5z7Uz0gUJZKQy8zYmz8YbgN0EoCuyeraK4E1aU9MnyOw6aFvrfw6YBut7L794aqky5HifyHe56vabkZycnUNEpJtzw02+rzP1i84snMR2dMnYiY5GL+QJAbbo1xvJhCU+83In8zPdxsfWQPSSA-Z82zvQZf+gT/8WvIhpNphk4JGB2bpr1lGDKwUaDw6hR/Wh/dfDFLUNMeOgQAH1mK8Cn7eJOw1tjx8hs+GQ97L1FpW7DX+9MY3MQDUK2NUmnj69x9nCMUtLeN+R+CU01aDZU8VAgR2iquJAZcGtfUtJS31K9iRFeelaRbtQKnaRtDVJS5JH6o19qixb315MEAXFHjdxx9fWjWrtAhj3R5imeFbFY132neb3w==

-----END RSA PRIVATE KEY-----

Certificate for b.example.com.

-----BEGIN CERTIFICATE-----
MIIDBzCCAkngeBGAgaIBATANBgkqhkiG9w0BAQUFAwIwMQswCQYDVQQHEwJVUzEiTMBEGA1UECMQKQ2FsaWVucpMBMIHJQVQGEmwCwYDVQQDBQJUZXN0IENlcnRpZmljYXRlIEZhY3Rpb24gQ29udGVudDAwDgYDVQQLDwJHbG9nYXRpb24gQ29udGVudDQwLjAAMIIBIwIjANBgkqhkiG9w0BAQsFAAOCAQEEKeu4ZlVitOEFp5LSUf1PjUO7b0P1VBIDxxwH/DS4rYf8iGyR0nK5iN5TkL/2oOAVXz2h5H7Rf/93oC1xW0U6i7bCLsKzP+cvsYm2JZzcR78wWKTdA0i3U6kGdA9V/Hk5YI6/AuBQ6QxYlVz0+0hZ5Gq68ct9lMN03G8msuFM9P9CJh6C0+F/9JgR3gH/3AzOj3lM0fV7hUvMEkC0gUa9dQhGvW0Jm8wOj4AMeLx8M4P7BnBcP6Q1+bc50ww4HJqy5wMVbJjZc6Hj9h+J6uE04UoC6oMNk1OgoJfF+G79dIc1R0Y3n50Fpc6g3u57sMw0mggQF1FHwJz3aCq6f0iuVnW5Pw5H0JAc4iUa7f7qTeh4H0H742SxwHjxZB9za7fSTs0X1uhY+2Uz90zDP2Alr6/pc0fmmSe9U3d4K7jQf1uo80jav2QcSUF12fZdE0R6pStez795q+xwE8/8hZfAe/6vPmTkk+IFkGBv094zDowbe3JNvN5h9Ew2tj/3XQ0kM/Tgr0JrKJj scraped text here

-----END CERTIFICATE-----

Private key for b.example.com.

-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQDQXnzK3my3is16QFX81IXwdeTEWvGxSAiHwhtfpsa+bZk3LQ6yPyTdETTkpdKzy5+GxxKc+7nOAzb3BwdpruvQ+aI8M/SDNFyghChXwqgNBMs5g7B1dGP1tICc/26GEMU1enHk7enqnsP97Tqnr94eh5K1jAUKQ0x2nF0bUwIAQABAoGAltyBLlLVmnWxDh31i887DY/Auo7Me22WnHxHQCtSMIPtQsbj9Rjz3J6b4y19lK5SghDwDdUvfrZxx7KXL3zbrdHuM2pQB7eR0mLiiQPkF7zX4FDFpDfjekufq7jPJehl7ZYH5GYPAzRQXa/SAX/NOSKMhQLYiwaJ5bRECCQD6zRL/vPAhAjC7OD0yFstTbdFgVzT2lGhnaAh9FpMiCSr7Uy7VRRr5z7Uz0gUJZKQy8zYmz8YbgN0EoCuyeraK4E1aU9MnyOw6aFvrfw6YBut7L794aqky5HifyHe56vabkZycnUNEpJtzw02+rzP1i84snMR2dMnYiY5GL+QJAbbo1xvJhCU+83In8zPdxsfWQPSSA-Z82zvQZf+gT/8WvIhpNphk4JGB2bpr1lGDKwUaDw6hR/Wh/dfDFLUNMeOgQAH1mK8Cn7eJOw1tjx8hs+GQ97L1FpW7DX+9MY3MQDUK2NUmnj69x9nCMUtLeN+R+CU01aDZU8VAgR2iquJAZcGtfUtJS31K9iRFeelaRbtQKnaRtDVJS5JH6o19qixb315MEAXFHjdxx9fWjWrtAhj3R5imeFbFY132neb3w==

-----END RSA PRIVATE KEY-----

Certificate for b.example.com.
19. Message Dumps

This section contains base64 encoded versions of the SIP messages in this draft. They can be encoded and used as test vectors, and they contain all the correct CRLF sequences. A command like "openssl base64 -d -in foo.b64 -out foo" will convert the base64 data to a SIP message that contains everything after the UDP header. This can be used with a net cat program like nc to send test messages to
The following is the base64 of the signed message.

```
TUVTU0FHRbZxaXA6Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
TUVTU0FHRbZxaXA6Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
TUVTU0FHRbZxaXA6Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
Ym9iQGIuZxhxbXBsZ5j2b0g0U10gLzuoMA0KVG86IDxzaXA6
```

This is the base64 of the signed message.
The base64 of the response was:

U01QLzIuMCAyMDAgT0sNC1lRoVoiA8c21wOmJyYkBilMvY4YW1wbGUuY29tPjt0YWc9
MzIwNjdiZDdkNzcyb206IDzixAXAYWxpY2VAYS5leGFtcGxlMnVzbT47dGFnPTFi
MmYiNzY5DQpWaWEFJNJC8yLiJvYVURQ1QDeyNy4wLiJyJmTM0IDcwO2JyYW5jaD16
OWHhJGJLMvNzU0M3ZwAwZnU0MDYMS0tZyZgNTQyLttecG9yID01MDcwOzJ1
Y2VpdmVkJrPeNy4wLiJyJmAuMQKQ2sbc1JRDogMjJiNjgyNMQmY2VyMzEwQzQKQ1N1
cTo9SMRsVNTQUDQpDb250YWN0I1A8c21wOmJyYkBilMvY4YW1wbGUuY29tOjUw
NjA+DQpDb250ZW50LUxldm0aDoMA0KDQo=

The following is the base64 of the encrypted message.

TUVTU0FHRBZaXA6Ym9i9IQGiuzXhhbXbsZ5j2b0gU01QLzIuMA0KVG86IDzAXA6
Ym9i9IQGiuzXhhbXbsZ5j2b0gDQpQcm90Ia8c21wOmFsaWNIQGExZhhbXbsZ5j
b20+03RhZz00YmJhMWYxZAZOkmhiO1BT5AVMi4wL1VEUCAxMjcuMC4wLjE6NTA3
MDticmFuY29jej0SRsiy1J0dCN1NDItNTU4NDIyODM0LTEtLWM4NzU0Mi01MDcwO3
qNCQnKnbGwtSUQ6IDZzeMmJ1ODk1MDE5ZDQ1MzYnY2h1NzKZXE6IDGEgDQpTU0FHRQ
Q29uODFjJdodPHNpBoBghiJLZmhLmV4YW1wbGUuY29tOjUwNzA+DQpNYXgtRm9y
dZ2FyZHMeIDcwQ29udGQwY29uZ2FyZHM6IDEwCg0KQ2FsbC1JRDogMTI3NzkyYz
au5wPXMjcxVzpcmK02pbgmYW11IPXntaW11Lm5Qpb250ZW50L0V5c0UGFw
cQxpY2F0aW9uL3BrY3M3LW1pbWU7c21pbWUtDHlwZTl1bn2bG9w92QtGZFGY7Tu
YW11PXntaw11LaA3b0KXVLNLlc1e2ZVydDqU01qaW11Ly92LyBjiMiAoY3Vy
c2VzQjQ0Q23udGVudC1NzW5ndGg6IDM4Q0QDQowggF9B9gkqkhk1G9wOBmQggu
M1IBaGJ1A0DCAyWyggESgEAMHswcDEJMcAgA1UEhhMCVVMxZEsAR5BGTVtKnh
bG1mb3JuaWExETAPB6NvBAcTCFvNlEiBKnq4wDADYVQKEwZaxMBpDDEpCnC
A1UEcxMjgu21waQxOgZVdCBDZXJOA2pYF0ZSBQxrob3pHkCB1UbBQJAHw1
QYJK0ZIhvcNQEAQEBQAggYGQ0SVKQ07kymKOSzSMpB0WBxI4YQ/1b9c52Lo5ze9+
mzthE+09Yf5iCzecZ208jYJ0LXasg81meW+RXXjLd9eOEKbcN2NmW3WVU1GNCk
Ubr31ICk4pP10M3C/+/qVj/6CRVBvQrteCE7ANC0W0hFbFue2aBfh1Uobe4a5Y2
qzqaLBgkQhlG9wOBmBwEfAY1kZlhcNAwEcJyY8c0GB9S9c1P8VFLeq5gT+X
pr4tR390sIz3dS3PvDH9SyOMELtStFhkKcdaL

The base64 of the response was:

U01QLzIuMCAyMDAgT0sNC1lRoVoiA8c21wOmJyYkBilMvY4YW1wbGUuY29tPjt0YWc9
MzIwNjdiZDdkNzcyb206IDzixAXAYWxpY2VAYS5leGFtcGxlMnVzbT47dGFnPTFi
YmExZjMkDQpWaWEFJNJC8yLiJvYVURQ1QDeyNy4wLiJyJmTM0IDcwO2JyYW5jaD16
OWHhJGJLMvNzU0M3ZwAwZnU0MDYMS0tZyZgNTQyLttecG9yID01MDcwOzJ1
Y2VpdmVkJrPeNy4wLiJyJmAuMQKQ2sbc1JRDogMjJiNjgyNMQmY2VyMzEwQzQKQ1N1
cTo9SMRsVNTQUDQpDb250YWN0I1A8c21wOmJyYkBilMvY4YW1wbGUuY29tOjUw
NjA+DQpDb250ZW50LUxldm0aDoMA0KDQo=

The following is the base64 of the encrypted message.

TUVTU0FHRBZaXA6Ym9i9IQGiuzXhhbXbsZ5j2b0gU01QLzIuMA0KVG86IDzAXA6
The base64 of the response was:

U0lQLzIuMCAyMDAgT0sNClRvOiA8c2lwOmJvYkBILmV4YW1wbGUuY29tPjt0YWc9
NDbakNzEzWINCkZyb206IDxzaXA6YWyxpY2VAY5seGftcGxlLmNvbT47dGFnPTFk
ODYzM2EzDQpWaWE6IFNjUC8yLjAvVURQIDEyNy4wLjAuMTo1MDcwO2JyYW5jaD16
OWhHNGJLWkMzU0Mi00ODg4ODQxMDQtMS0tYzg3NTQyLzg3NTQyLTycG9ydG1lcmw3Jl
Y2VpdmVkPTEyNy4wLjAuMQU0KQ2FsbcClJRdogNDUwYzhilMTEyNjE1YzczMg0KQ2Fsb
ctogMSBNRVNTQUdFDQpDb250YW1wbGUuY29tOjUw

20. Open Issues

Should the type of the subjectAltName be URI or RFC822 in S/MIME examples?

Is the encrypted and signed example in this draft correct with respect to what the signature in a detached signature is computed over?

The examples here attach the sender’s certificates – is this how we want to go? Need more text on when or should or should not do this.

Need to added Accept with multipart to all examples. Might also want to request congestion safety on all of them.

21. Still To Do

The examples here attached the sender’s certificates – is that the way we want to go. Need more text on when or should or should not do this.

Examples showing keywrap stuff.

Examples in 3261.

Would be nice to add example showing encrypted SDP with SRTP key examples.

Would be nice to add example showing securing a refer.

Jennings
Expires April 18, 2004
22. Acknowledgments

Many thanks to the developers of all the open source software used to create these call flows. This includes the underlying crypto and TLS software used from openssl.org, the SIP stack from www.resiprocate.org, and the SIMPLE IMPP agent from www.sipimp.org. The TLS flow dumps were done with SS LDump from http://www.rtfm.com/ssldump. The book SSL and TLS [9] was a huge help in developing the code for these flows and is a great resource for anyone trying to implement TLS with SIP.

Thanks to Dan Wing and Robert Sparks for catching many silly mistakes and Tat Chan who caught a key problem in what the signature was being computed over.

Normative References


Informative References


Author’s Address

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/3
San Jose, CA 95134
USA

Phone: +1 408 527-9132
EMail: fluffy@cisco.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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