Internet Engineering Task Force
Internet Draft
draft-ietf-mmusic-sip-sec-01.txt
March 12th 1998
Expires: September 12th 1998

MMUSIC WG
Peter Kirstein
Goli Montasser-Kohsari
Edmund Whelan
University College London

SIP Security Using Public Key Algorithms

Status of this Memo

This document is an Internet-Draft. 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.”

To learn the current status of any Internet-Draft, please check the ‘‘1id-abstracts.txt” listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

Abstract

The Session Initiation Protocol (SIP) is a simple protocol designed to enable the invitation of users to multimedia sessions. This document is a companion draft to draft-ietf-mmusic-sip-04.txt, which defines SIP but doesn’t specify any security mechanisms other than possible protection by lower-level security mechanisms such as SSL. This leaves SIP transactions vulnerable to attack and this document aims to extend the SIP protocol to address such security considerations.

This document is a product of the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force Comments are solicited and should be addressed to the working group's mailing list at confctrl@isi.edu and/or the authors.

Table of Contents

1. Introduction

1.1 Overview of SIP Functionality

The Session Initiation Protocol (SIP) is an application-layer protocol that can establish and control multimedia sessions or calls. These multimedia sessions include multimedia conferences, distance learning, Internet telephony and similar applications. SIP can invite a person to both unicast and multicast sessions; the initiator does not necessarily have to be a member of the session it is inviting to. Media and participants can be added to an existing session. SIP can be used to "call" both persons and "robots", for example, to invite a media storage device to record an ongoing conference or to invite a video-on-demand server to play a video into a conference. SIP can be used to initiate sessions as well as invite members to sessions that have been advertised and established by other means such as SAP [1], electronic mail, news groups, web pages and directories (LDAP). SIP is fully defined in [2] and, as this is a companion draft aiming to extend the basic functionality of SIP to address security considerations, familiarity with [2] is assumed. Consequently, only a brief introduction to SIP itself is included here.

1.2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [28] and indicate requirement levels for compliant SIP implementations.

1.3 SIP Message Overview

As much of the message syntax is identical to HTTP/1.1 the notation [HX.Y] is used to refer to section X.Y of the current HTTP/1.1 [16] specification. In addition an augmented Backus-Naur form (BNF)[17, H2.1] is used. All SIP messages are text-based and use HTTP/1.1 conventions [H4.1], except for the additional ability of SIP to use UDP. When sent over TCP or UDP, multiple SIP transactions can be carried in a single TCP connection or UDP datagram. UDP datagrams, including all headers, should not normally be larger than the path maximum transmission unit (MTU) if the MTU is known or 1500 bytes if the MTU is unknown. The 1400 bytes accommodates lower-layer packet headers within the "typical" MTU of around 1500 bytes. There are few MTU values around 1 kB; the next value is 1006 bytes for SLIP and 296 for low-delay PPP [18]. Recent studies [19] indicate that an MTU of 1500 bytes is a reasonable assumption. Thus, another reasonable value would be a message size of 950 bytes, to accommodate packet headers within the SLIP MTU without fragmentation.

A SIP message is either a request from a client to a server, or a response from a server to a client.

```
SIP-message      =    Request | Response ; SIP messages
```

Both Request and Response messages use the generic RFC822 [3] message format for transferring entities (the payload of the message). Both types of message consist of a start-line, one or more header fields, an empty line indicating the end of the header fields and an optional message body.

```
Generic message  =    start-line
                      *(general header | request header | entity header)
                      CRLF
```

[message body]

start-line = Request Line | Status Line

1.3.1 Request

The Request message format is shown below:

```
Request      =   Request Line
                 *(general header | request header | entity header)
                 CRLF
                 [message body]
```

The Request Line begins with a method token, followed by the Request-URI, the protocol version and ending with a CRLF. SP characters separate each element.

```
Request Line   =   Method SP Request-URI SP SIP-Version CRLF

Method       =   "INVITE" | "ACK" | "OPTIONS" | "BYE" | "REGISTER" | "UNREGISTER"`
```

INVITE indicates that the user or service is being invited to participate and the message body contains a description of the relevant session. ACK confirms that the client has received a final response to an INVITE request. OPTIONS is used to query a client about its capabilities. The client uses BYE to indicate to the server that it wishes to abort the call attempt. REGISTER is used by the client to register the address listed in the request line to a SIP server and UNREGISTER is used to cancel an existing registration. Support for INVITE and ACK is mandatory in the SIP specification whereas support for OPTIONS, BYE, REGISTER and UNREGISTER is OPTIONAL.

1.3.2 Response

Once a recipient has received and interpreted a request message they respond with a SIP response message. The format of the response is shown below:

```
Response     =   Status Line
                 *(general header | request header | entity header)
                 CRLF
                 [message body]
```

The Status Line is similar to the Request Line. It consists of the SIP Version, a numeric Status Code and its associated textual phrase. SP characters separate each element and the line ends with a CRLF sequence.

```
Status Line    =   SIP Version SP Status Code SP Reason Phrase CRLF
```

1.3.3 SIP Version

Both request and response messages include the version of SIP in use and this follows [H3.1], with HTTP replaced by SIP. Applications making use of the security enhancements detailed in this paper MUST include a SIP-Version of "SIP/2.1".

1.4 SIP Transaction

When making a SIP call a caller first locates the appropriate sever and then sends a SIP request. The most common SIP operation is the invitation. Instead of directly reaching the intended callee, a SIP request may be redirected or trigger a new chain of requests by proxies.

1.4.1 Example of a SIP Invitation

A successful SIP invitation consists of two requests, INVITE followed by ACK. The INVITE request asks the callee to join a particular conference or establish a two-party conversation. After the callee has agreed to participate in the call, the caller confirms that it has received that response by sending an ACK request. If the call is rejected, or otherwise unsuccessful, the caller sends a BYE request instead of an ACK.

The INVITE request typically contains a session description, for example written in SDP format [4], that provides the called party with enough information to join the session. For multicast sessions, the session description enumerates the media types and formats that may be distributed to that session. For unicast sessions, the session description enumerates the media types and formats that the caller is willing to receive and where it wishes the media data to be sent. In either case, if the callee wishes to accept the call, it responds to the invitation by returning a similar description listing the media it wishes to receive. For a multicast session, the callee should only return a session description if it is unable to receive the media indicated in the caller's description. The caller may ignore the session description returned or use it to change the global session description.

The protocol exchanges for the INVITE method are shown in Fig. 1 for a proxy server and in Fig. 2 for a redirect server. The proxy server accepts the INVITE request (1), contacts the location service with all or parts of the address (2) and obtains a more precise location (3). The proxy server then issues a SIP INVITE request to the address(es) returned by the location service (4). The user agent server alerts the user (5) and returns a success indication to the proxy server (6). The proxy server then returns the success result to the original caller (7). The receipt of this message is confirmed by the caller using an ACK message, which is forwarded to the callee (8,9), with a response returned (10,11). All requests have the same Call-ID.

```
                                         +....... cs.columbia.edu .......+
                                         :                               :
    ===> SIP Request                     : (~~~~~~~~~~)                  :
    ---> non-SIP Protocols               : ( location )                  :
                                         : ( service  )                  :
                                         : (~~~~~~~~~~)                  :
                                         :   ^      |                    :
                                         :   |   hgs@play                :
                                         :  2|     3|                    :
                                         :   |      |                    :
                                         : henning  |                    :
+.. cs.tu-berlin.de ..+ 1: INVITE        :   |      |                    :
:                     :    henning@cs.col:   |      | 4: INVITE  5: ring :
: cz@cs.tu-berlin.de ========================> tune  =========> play     :
:                    <........................       <.........          :
:                     : 7: 200 OK        :            6: 200 OK          :
+.....................+                  +...............................+
```

Figure 1: Example of SIP proxy server

The redirect server accepts the INVITE request (1), contacts the location service as before (2,3) and, instead of contacting the newly found address itself, returns the address to the caller (4). The caller issues a new request, with a new call-ID, to the address returned by the first server (6). In the example, the call succeeds (7). The caller and callee complete the handshake with an ACK (8,9).

The action taken on receiving a list of locations varies with the type of SIP server. A SIP redirect server simply returns the list to the client sending the request as Location headers. A SIP proxy server can sequentially or in parallel try the addresses.

```
                                         +....... cs.columbia.edu .......+
                                         :                               :
    ===> SIP Request                     : (~~~~~~~~~~)                  :
    ---> non-SIP Protocols               : ( location )                  :
                                         : ( service  )                  :
                                         : (~~~~~~~~~~)                  :
                                         :   ^      |                    :
                                         :   |   hgs@play                :
                                         :  2|     3|                    :
                                         :   |      |                    :
                                         : henning  |                    :
+.. cs.tu-berlin.de ..+ 1: INVITE        :   |      |                    :
:                     :    henning@cs.col:   |      |                    :
: cz@cs.tu-berlin.de =======================>  tune                      :
:         ^ |        <.......................                            :
:         . |         : 4: 302 Moved     :                               :
+.........|.........+    hgs@play        :                               :
          . |                            :                               :
          . | 5: INVITE hgs@play.cs.columbia.edu               6: ring   :
          . ==================================================> play      :
          .....................................................          :
            7: 200 OK                    :                               :
                                         +...............................+
```

Figure 2: Example of SIP redirect server

If a proxy server forwards a SIP request, it MUST add itself to the end of the list of forwarders noted in the *Via* headers. The *Via* trace ensures that replies can take the same path back, thus ensuring correct operation through compliant firewalls and loop-free requests. On the reply path, each host most remove its *Via*, so that routing internal information is hidden from the callee and outside networks. When a multicast request is made, first the host making the request, then the multicast address itself are added to the path. A proxy server MUST check that it does not generate a request to a host listed in the *Via* list. (Note: If a host has several names or network addresses, this may not always work. Thus, each host also checks if it is part of the *Via* list.)

1.5 Transport-Protocol

SIP is able to utilise both UDP and TCP as transport protocols. UDP allows the application to more carefully control the timing of messages and their retransmission, to perform parallel searches without requiring TCP connection state for each outstanding request, and to use multicast. Routers can more readily snoop SIP UDP packets. TCP allows easier passage through existing firewalls, and given the similar protocol design, allows common servers for SIP, HTTP and the RTSP [5].

2. Security Considerations

The basic SIP Draft [2] does not deal with security considerations other than to specify a reliance on lower-layer security mechanisms such as SSL [6] when this is required. However, a number of issues need to be addressed. These include the following:

- **Authentication and Integrity of the SIP Communications**

  Here it is necessary, for example, to ensure that if a callee receives an invitation they can be confident that the invitation did indeed originate from the claimed initiator and has not been changed *en route*. SIP messages can also modify details of future conferences and so it would also be possible for a *denial of service* attack to be successful if messages cannot be authenticated.

- **Confidentiality of Conference Details**

  Here it is at least necessary to hide the details of the addresses and media formats used. However, some proxies need to read certain SIP header fields.

In order to facilitate the widespread implementation and use of SIP security it was considered important to re-use existing security implementations as much as possible.

2.1 Symmetric and Asymmetric Encryption

The simplest versions of encryption are *symmetric* ones; here the encryption key can be calculated from the decryption key and vice versa. In most cases the encryption key and decryption key are the same. This means that, if E{a,M} is the operation of encrypting the message M with the key *a* and algorithm E, then the decryption operation D{*a*, E{*a*, M}} reproduces the original message M. If several people know the key then symmetric encryption cannot be used for authentication.

An alternative form of encryption is with the use of *asymmetric* algorithms (also known as Public Key algorithms). Here the key used for encryption is different to that used for decryption and it is not feasible to calculate one from the other. Consequently the encryption key can be made public and is known as the Public Key. Encrypting a message with the recipient’s Public Key ensures confidentiality as only the recipient with the corresponding decryption key (known as the Private Key) can decrypt the message. Encrypting a message with the Private Key of the sender ensures authentication as only the sender could have sent the message whereas anybody having access to the Public Key can verify that it was indeed sent by the person holding the corresponding Private Key. Some Public Key algorithms (e.g. RSA [7]) can be used for both digital signatures and encryption whereas others (e.g. DSA) can only be used for digital signatures.

Most practical implementations of public key cryptography use a combination of symmetric and asymmetric algorithms. This is largely due to the fact that symmetric algorithms are generally much faster than asymmetric ones as well as the fact that public key cryptosystems are vulnerable to chosen-plaintext attacks. Consequently, the messages are generally secured using a symmetric algorithm and a different session key each time. This session key is then encrypted and distributed using public key algorithms.

The two public key algorithm formats which we make use of in this document are PGP [8] and S/MIME [9]. These can be used for both encryption and authentication. As detailed later implementers MUST support PGP formats with support for S/MIME being OPTIONAL.

3. SIP Security with PGP and MIME

The method specified here to secure SIP with PGP and MIME draws on RFC 1847 [10] and RFC 2015 [11].

3.1 PGP Data Formats

PGP can generate either ASCII armour [8] or 8-bit binary output when encrypting data, generating a digital signature or extracting a public key. With SIP an 8-bit clear channel is available and so ASCII armour is not necessary. However, for interoperability issues receiving agents SHOULD support the receipt of both armoured and unarmoured data. Receiving and sending agents MUST support unarmoured data.

3.2 MIME Encapsulation

Prior to any security procedures the SIP message MUST be converted into MIME canonical format. The procedure for creating a MIME entity is given in [20] where the reader is referred for fuller details than are provided here. As SIP messages are text-based the MIME canonicalization for the type text/plain should be followed. In brief the line ending must be the <CR><LF> sequence and the charset should be a registered charset [21]. The details are specified in [20] and the chosen charset SHOULD be named in the charset parameter so the receiving agent can unambiguously determine the charset used.

As SIP has an 8-bit clear channel available then no transfer encoding is necessary. However this should be supported in order to interoperate more easily and allow messages created with mail tools etc to be used without change. If a multipart/signed entity is being used it SHOULD have transfer encoding applied so that it is represented as 7-bit text. This would allow the signature on the data to be identical if it had been sent via email or via SIP which enhances flexibility.

An example of a MIME prepared SIP message, with no security enhancements added, follows:

```
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

INVITE j.doe@acme.com SIP/2.1
To: J.Doe <joe@acme.com>
From: Michael Elkins <elkins@aero.org>
Subject: Videoconference on security issues
CALL-ID: 786598
Content-Type: application/sdp

v=0
o=anybody 3098086191 3098086197 IN IP4 anycomputer.anywhere.com
s=general conference
i=This is a general conference
e=A. Nyone <anyone@anywhere.com>
p=A. Nyone +44 171 123 4567
c=IN IP4 239.255.140.52/15
```

```
t=3098084400 3098091600
a=tool:sdr v2.5a0
a=type:test
m=audio 19274 RTP/AVP 0
c=IN IP4 239.255.140.52/15
a=ptime:40
```

Outside the MIME body the only SIP headers used are those dealing with routing information which the proxies need access to. These include the *From, To* and *Call-ID* fields. In addition, if any other fields such as *Proxy-Authenticate* or *Proxy-Authorization* are present in the SIP message these SHOULD also appear outside the MIME body if the SIP message is only signed and MUST be present when the message is encrypted. This allows sensitive fields such as the *Subject* field to be present only in the MIME body which, when encrypted, makes these unreadable by the proxies. The SIP-header Content-Type MUST NOT appear with the headers outside the MIME body as this header field name is used by MIME.

A “MIME-Version” header field [20] is then appended to the top-level SIP headers and this must include the following text:

MIME-Version: 1.0

This indicates that the SIP Messages have been produced in accordance with [20]. Note that all header fields in this document follow the general syntactic rules defined in RFC 822 and in particular comments are allowed. This means that the following are equivalent.

MIME-Version: 1.0
MIME-Version: 1.0 (Generated by SIP Server 3.4)

3.3 PGP Encrypted Data

PGP encrypted data is denoted by the “multipart/encrypted” content type, as in RFC 1847, and MUST have a *protocol* parameter value of “application/pgp-encrypted” including the quotes. The multipart/encrypted message MUST consist of exactly two parts. The first MIME body part MUST have a content-type of “application/pgp-encrypted” and contain the control information. The PGP message itself contains all the relevant control information so a message complying with this standard MUST contain a “Version: 1” field in this body. The second MIME body part MUST contain the actual encrypted data and be labelled “application/octet-stream”. The PGP message is the complete, original SIP message converted into MIME canonical format as in Section 3.2 and encrypted with the recipient’s public key. An example of an invitation where the payload of the SIP message is in SDP format and ASCII armour has been used follows:

```
INVITE j.doe@acme.com SIP/2.1
From: John Smith <J.Smith@aa.org>
To: J.Doe <joe@acme.com>
Call-ID: 554565
MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary=foo;
     protocol=”application/pgp-encrypted”

--foo
```

```
Content-Type: application/pgp-encrypted

Version: 1

--foo

Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: 2.6.3i
hIwDjHVjzUcPoL0BA/0TlaOlL9qyUoEqP+jvSBKZLkQD9vU14O4WWf9h0EPUnHwQ
6f8r+9YaxPvU7a25Oct2QE2oOIr+smDr+Z/NKld58ueFqvNQ+kXac9vRAzv403vt
YYrh4ug1eetpe8znskWHiJv+3LHQBsNyWwLwoC78rIM2KQomhNUC6nvD9BP03qYA
AADXS6iYIRljShpEtc137hBhx4Q9sXAh2ki2OnEPom+q0Pqwy2C5SgZfHfotg9Pt
3HWRSJ64Se0VN2SQl9jFZ0oAkKtY79sCvkyhRGIjteLlufNbGBbLnbghN7uUBQi/
URhXSWybyVMc2NoQaivRQxyHcXvhs8/687qPG2tGParPro2dbk8HiUN88lcdUCuL
+aD8qDllVhESVkdXfUHvL28fHhdM2ERaFCKQVOt9/7mnF3LB6SSbJKZuZJ+BYg/3
zuBVqDZo49yfURsgZ/iU2HWMXUydYEvxOR8=
=1I+h
-----END PGP MESSAGE-----

--foo--
```

Upon receipt of a multipart/encrypted body part the following is done. The second body part is prepared for decryption according to the value of the protocol parameter The prepared body part is made available to the decryption process and this makes available the result of the decryption and the decrypted form of the encrypted body part. The SIP implementation then continues processing with the decrypted body part. The unencrypted headers outside the MIME body SHOULD be discarded to avoid unwanted additions apart from the *Via* headers, added by the intermediate proxies, which are needed for routing any replies.

The specification of a new SEALED method whereby the entire contents of the SIP message, including the *From, To* and *Call-ID* fields, are encrypted with the public key of the server at the next hop was considered. This has the advantage that all sensitive information is encrypted during transit from server to server but has the disadvantage that each server can gain access to the full conference details etc. This is clearly undesirable and the possibility of encrypting a message as above with the recipient's public key and then re-encrypting it with the public key of the server at the next hop would be more secure. This allows each server access only to the information needed. However, this gives rise to a considerable overhead in both message size and processing time and is considered to give minimal benefit. Traffic analysis could gain access to information similar to that contained in the *From* and *To* fields and the other sensitive fields such as *Subject* are already encrypted with the method detailed above.

3.4 PGP Signed Data

PGP signed messages are denoted by the "multipart/signed" content type as in RFC 1847 with a protocol parameter which MUST have a value of "application/pgp-signature". The "micalg" parameter MUST have a value of "pgp-<hash-symbol>" where <hash-symbol> identifies the message integrity check used to generate the signature. The currently defined values for <hash-symbol> are "md5" for the MD5 checksum and "sha1" for the SHA.1 algorithm.

The multipart/signed message MUST consist of exactly two parts, with the first part containing the data to be signed, in MIME canonical format, including a set of appropriate content headers describing the data. The second body part MUST contain the PGP digital signature and MUST be labelled with a content-type of “application/pgp-signature”. The message is created in the same way as above in that only the information relevant to the proxies is outside the PGP signed message. As described in [10], the digital signature MUST be calculated over both the data to be signed and the set of content headers belonging to the body part to be signed. Also the signature MUST be generated detached from the signed data so that the process does not alter the signed data in any way.

An example of a PGP signed SIP message would be:

```
INVITE j.doe@acme.com SIP/2.1
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
CALL-ID: 786598
Mime-Version: 1.0
Content-Type: multipart/signed; boundary=bar; micalg=pgp-md5;
     protocol="application/pgp-signature"

--bar
> Content-Type: text/plain; charset=iso-8859-1
> Content-Transfer-Encoding: quoted-printable
>
> INVITE j.doe@acme.com SIP/2.1
> To: J.Doe <joe@acme.com>
> From: Michael Elkins <elkins@aero.org>
> Subject: Videoconference on security issues
> CALL-ID: 786598
>
> Content-Type: application/sdp
>
> v=0
> o=anybody 3098086191 3098086197 IN IP4 anycomputer.anywhere.com
> s=general conference
> i=This is a general conference
> e=A. Nyone <anyone@anywhere.com>
> p=A. Nyone +44 171 123 4567
> c=IN IP4 239.255.140.52/15
> t=3098084400 3098091600
> a=tool:sdr v2.5a0
> a=type:test
> m=audio 19274 RTP/AVP 0
> c=IN IP4 239.255.140.52/15
> a=ptime:40
>
--bar
Content-Type: application/pgp-signature

-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//
jJV5bNvkZIGPIcEmI5iFd9boEgvpirHtIREEqLQRkYNoBActFBZmh9GC3C041WGq
uMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9Brn
HOxEa44b+EI=
=ndaj
-----END PGP MESSAGE-----

--bar--
```

The ">"s in the previous example indicate the portion of the data over which the signature was calculated. Upon receipt of a signed message, an application MUST pass both the signed data and its associated content headers along with the PGP signature to the signature verification service.

3.5 PGP Encrypted and Signed Data

With SIP messages it is often desirable to digitally sign and then encrypt the messages. There are two methods for accomplishing this [11]. The data can be both encrypted and then signed or can undergo a joint “encryption and signing” process. Sending agents MUST support the joint process. Receiving agents SHOULD support both types of encryption and signing.

3.5.1 RFC 1847 Encapsulation

Here, the data (SIP Message) should first be signed as a multipart/signed body as before and then encrypted to form the final multipart/encrypted body as below.

```
 INVITE j.doe@acme.com SIP/2.1
 From: John Smith <J.Smith@aa.org>
 To: J.Doe <joe@acme.com>
 Call-ID: 554565
 MIME-Version: 1.0
 Content-Type: multipart/encrypted;
    protocol="application/pgp-encrypted"; boundary=foo

--foo
Content-Type: application/pgp-encrypted

Version: 1

--foo
Content-Type: application/octet-stream

 -----BEGIN PGP MESSAGE-----
 > Content-Type: multipart/signed; boundary=bar; micalg=pgp-md5;
 >          protocol="application/pgp-signature"
 >
 > --bar
 > Content-Type: text/plain; charset=iso-8859-1
 >
 > INVITE j.doe@acme.com SIP/2.1
 > From: Michael Elkins <elkins@aero.org>
 > To: J.Doe <joe@acme.com>
 > CALL-ID: 786598
 > Content-Type: application/sdp
 >
 > v=0
 > o=anybody 3098086191 3098086197 IN IP4 anycomputer.anywhere.com
 > s=general conference
 > i=This is a general conference
 > e=A. Nyone <anyone@anywhere.com>
 > p=A. Nyone +44 171 123 4567
 > c=IN IP4 239.255.140.52/15
 > t=3098084400 3098091600
 > a=tool:sdr v2.5a0
 > a=type:test
 > m=audio 19274 RTP/AVP 0
```

```
 > c=IN IP4 239.255.140.52/15
 > a=ptime:40
 >
 > --bar
 > Content-Type: application/pgp-signature
 >
 > -----BEGIN PGP MESSAGE-----
 > Version: 2.6.3i
 >
 > iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//
 > jJV5bNvkZIGPIcEmI5iFd9boEgvpirHtIREEqLQRkYNoBActFBZmh9GC3C041WGq
 > uMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9Brn
 > HOxEa44b+EI=
 > =ndaj
 > -----END PGP MESSAGE-----
 >
 > --bar--
-----END PGP MESSAGE-----

--foo--
```

The text preceded by '>' indicates that it is really encrypted, but presented as text for clarity.

3.5.2 Combined Encryption and Signing

PGP also allows data to be signed and encrypted in one operation. As mentioned above both sending and receiving agents MUST support this format and this joint method has less overhead in terms of both data length and processing time. The resulting data should be formed as a "multipart/encrypted" object as described in Section 2.3. Messages, which are encrypted and signed in this combined fashion, are REQUIRED to follow the same canonicalization rules as for multipart/signed objects.

Although only INVITE SIP messages have been used as examples in this document the format is applied in exactly the same way for the other method tokens mentioned above. This allows signature of ACK, BYE *etc* SIP messages.

3.6 PGP Supported Algorithms

In order to maintain wide interoperability the algorithms supported here follow [8] where fuller details can be found.

1. **Public Key Algorithms** - Implementations MUST implement DSA for signatures and Elgamal for encryption. Implementations SHOULD implement RSA encryption.

2. **Symmetric Key Algorithm** - Implementations MUST implement Triple-DES. Implementations SHOULD implement IDEA and CAST5.Implementations MAY implement any other algorithm.

3. **Compression Algorithms** - Implementations MUST implement uncompressed data. Implementations SHOULD implement ZIP.

4. **Hash Algorithms** - Implementations MUST implement SHA-1. Implementations SHOULD implement MD5.

4. SIP Security with S/MIME

Support for the use of the S/MIME message format is OPTIONAL in this specification.

The Cryptographic Message Syntax (CMS) is derived from PKCS#7 [14] and is fully defined in [15]. This syntax is used here to digitally sign or encrypt SIP messages and describes an encapsulation syntax for data protection. The CMS values are generated using ASN.1 [12] using BER encoding [13] and values are generally represented as octet strings.

CMS associates a protection content type with a protection content and has ASN.1 type ContentInfo:

```
ContentInfo   ::=  SEQUENCE {
     ContentType   ContentType,
     Content[0]    EXPLICIT ANY DEFINED BY contentType }

ContentType   ::=  OBJECT IDENTIFIER
```

ContentType indicates the type of protection content and is an Object Identifier. Six are defined in [15] but only Data, SignedData and EnvelopedData are of relevance to this document. Content is the actual protected content. Sending agents MUST use the “data” content type as the content within other content types to indicate the message content which has had security services applied to it. In addition sending agents MUST use the signedData content type to apply a digital signature to a message and an envelopedData type to encrypt a message. Further details of the specific requirements can be found in [9].

Prior to any security enhancements the SIP message is converted into MIME canonical format exactly as described in [20] and discussed in Section 3.2. When S/MIME formats are used then these are as defined in [9,15] where more complete details can be found. These documents are, in turn, based on RFC 1847 [10] and PKCS#7 [14].

4.1 The application/pkcs7-mime Type

The application/pkcs-7-mime type is used to carry CMS objects. This always carries a single CMS object which is the BER encoding of the ASN.1 syntax describing the object. The MIME headers of all application/pkcs-7-mime objects SHOULD include the optional ”smime-type” parameters as described in [9].

4.2 Encryption Only with S/MIME

The SIP message, which has been converted to MIME canonical format, is processed into a CMS [15] object of type envelopedData. This is then inserted into an applicaton/pkcs7-mime entity. The smime-type parameter for enveloped-only messages is “enveloped-data” and the file extension for this type of message is “.p7m”.

A sample message would be:

```
INVITE j.doe@acme.com SIP/2.1
```

```
From: John Smith <J.Smith@aa.org>
To: J.Doe <joe@acme.com>
Call-ID: 554565
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

4.3 Signing Only With S/MIME

There are two possible formats for signed messages. One can either use application/pkcs7-mime and SignedData or multipart/signed. In general, the multipart/signed form is preferable for sending, and receiving agents SHOULD be able to handle both. The receiver can always view messages signed using the multipart/signed format whether they have S/MIME software or not. A recipient cannot view messages signed using the signedData format unless they have S/MIME facilities.

4.3.1 Signing Using application/pkcs7-mime and SignedData

This signing format uses the application/pkcs7-mime MIME type as above but now the MIME entity and other required data is processed into a CMS object of type signedData. This is then inserted into an application/pkcs7-mime MIME entity. The smime-type parameter for messages using application/pkcs7-mime and SignedData is "signed-data" and the file extension for this type of message is ".p7m".

A sample message would be:

```
INVITE j.doe@acme.com SIP/2.1
From: John Smith <J.Smith@aa.org>
To: J.Doe <joe@acme.com>
Call-ID: 554565
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=signed-data;
     name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

4.3.2 Signing using the multipart/signed Format

This format is a clear-signing format and recipients with no S/MIME or PKCS processing facilities are able to view the message. It makes use of the multipart/signed MIME type described in RFC 1847 and the format is similar to the PGP signed-only SIP messages earlier. The multipart/signed MIME type has two parts. The first part contains the MIME entity that is to be signed; the second part contains the signature, which is a CMS detached signature.

In order to create this type of message the SIP message is again converted to MIME canonical format as discussed in Section 3.2. This MIME entity is then processed to obtain a CMS object of type signedData with an empty contentInfo field. The MIME entity is then inserted into the first part of a multipart/signed message. Following this, transfer-encoding can, if desired, be applied to the detached signature and this is inserted into a MIME entity of type application/pkcs7-signature. This MIME entity is then inserted into the second part of the multipart/signed entity.

The application/pkcs7-signature MIME type always contains a single CMS object of type signedData. The contentInfo field of the CMS object must be empty. The signerInfos field contains the signatures for the MIME entity and the file extension for signed-only messages using application/pkcs7-signature is ".p7s".

The multipart/signed content-type has two required parameters: the protocol parameter and the micalg parameter. The protocol parameter MUST be "application/pkcs7-signature". Note that quotation marks are required around the protocol parameter because MIME requires that the "/" character in the parameter value MUST be quoted. The micalg parameter allows for one-pass processing when the signature is being verified. The value of the micalg parameter is dependent on the message digest algorithm used in the calculation of the Message Integrity Check. The value of the micalg parameter SHOULD be one of the following:

```
Algorithm:    MD5      SHA-1
Value:        md5      sha1
```

An example of a multipart/signed SIP Invitation would be:

```
INVITE j.doe@acme.com SIP/2.1
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
CALL-ID: 786598
Mime-Version: 1.0
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=boundary42

--boundary42
> Content-Type: text/plain; charset=iso-8859-1
> Content-Transfer-Encoding: quoted-printable
>
> INVITE j.doe@acme.com SIP/2.1
> To: J.Doe <joe@acme.com>
> From: Michael Elkins <elkins@aero.org>
> Subject: Videoconference on security issues
> CALL-ID: 786598
>
> Content-Type: application/sdp
>
> v=0
> o=anybody 3098086191 3098086197 IN IP4 anycomputer.anywhere.com
> s=general conference
> i=This is a general conference
> e=A. Nyone <anyone@anywhere.com>
> p=A. Nyone +44 171 123 4567
```

```
> c=IN IP4 239.255.140.52/15
> t=3098084400 3098091600
> a=tool:sdr v2.5a0
> a=type:test
> m=audio 19274 RTP/AVP 0
> c=IN IP4 239.255.140.52/15
> a=ptime:40
>
 --boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

The “>” indicate the data included in the signature.

4.4 Signing and Encrypting

To achieve signing and encrypting, any of the signed-only and encrypted-only formats may be nested. This is allowed because the above formats are all MIME entities, and because they all secure MIME entities. A secure SIP implementation MUST be able to receive and process arbitrarily nested S/MIME within reasonable resource limits of the recipient computer.

It is possible to either sign a message first, or to envelope the message first. It is up to the implementor and the user to choose. When signing first, the enveloping then securely obscures the signatories. When enveloping first the signatories are exposed, but it is possible to verify signatures without removing the enveloping. This may be useful in an environment where automatic signature verification is desired, as no private key material is required to verify a signature.

4.5 S/MIME Supported Algorithms

In order to maintain wide interoperability the algorithms supported here follow [9] where fuller details can be found.

1. **KeyEncryptionAlgorithmIdentifier**

   Sending and receiving agents MUST support Diffie-Hellman defined in [22].

   Receiving agents SHOULD support rsaEncryption [7]. Incoming encrypted messages contain symmetric keys which are to be decrypted with a user’s private key. The size of the private key is determined during key generation.

   Sending agents SHOULD support rsaEncryption. If an agent supports rsaEncryption, then it MUST support encryption of symmetric keys with RSA public keys at key sizes from 512 bits to 1024 bits.

2. **SignatureAlgorithmIdentifier**

Sending and receiving agents MUST support id-dsa defined in [23]. The algorithm parameters MUST be absent (not encoded as NULL).

Receiving agents SHOULD support rsaEncryption, defined in [7]. Receiving agents SHOULD support verification of signatures using RSA public key sizes from 512 bits to 1024 bits.

Sending agents SHOULD support rsaEncryption. Outgoing messages are signed with a user's private key. The size of the private key is determined during key generation.

3. **ContentEncryptionAlgorithmIdentifier**

Receiving agents MUST support encryption and decryption with DES EDE3 CBC [24]. Receiving agents SHOULD support encryption and decryption using the RC2 [25] or a compatible algorithm at a key size of 40 bits.

4. **DigestAlgorithmIdentifier**

Receiving agents MUST support SHA-1 [26]. Receiving agents SHOULD support MD5 [27]. Sending agents SHOULD use SHA-1.

References

[1] M.Handley "SAP: Session Announcement Protocol"
Internet Draft: draft-ietf-mmusic-sap-00.txt, 27/Nov/96

[2] M. Handley, H. Schulzrinne and E. Schooler "SIP: Session Initiation Protocol"
Internet Draft: draft-ietf-mmusic-sip-04.txt, Nov 97

[3] D. Crocker "Standard for the Format of ARPA Internet Text Messages"
STD 11, RFC 822, University of Delaware, August 1982

[4] M. Handley and V. Jacobson "SDP: Session Description Protocol"
Internet Draft: draft-ietf-mmusic-sdp-06.txt, Jan 97

[5] H. Schulzrinne, A. Rao and R. Lanphier "Real Time Streaming Protocol (RTSP)"
Internet Engineering Task Force, Internet Draft, July 97 Work In Progress

[6] A. Freier, P. Karlton and P. Kocher "The SSL Protocol: Version 3.0"
Internet Draft: draft-ietf-tls-ssl-version3-00.txt, Nov 96

[7] R.L. Rivest, A. Shamir and L.M. Adleman "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM,* v. 21, n. 2, Feb 1978, pp 120-126

PKCS#1 "RSA Encryption Standard"
RSA Laboratories, Version 1.5, Nov 1993

[8] J. Callas "OpenPGP Message Formats"
Internet Draft: draft-ietf-openpgp-formats-00.txt, Nov 1998

[9] B. Ramsdell "S/MIME Version 3 Message Specification"
Internet Draft: draft-ietf-smime-msg-01.txt, Jan 1998

[10] J. Galvin et al "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted"
Technical Report RFC 1847, Internet Engineering Task Force, Oct 1995

[11] M. Elkins "MIME Security With Pretty Good Privacy"
Technical Report RFC 2015, Internet Engineering Task Force, Oct 1996

[12] X.208 "Specification of Abstract Syntax Notation One (ASN.1)"
ITU-T Recommendations, 1988

[13] X.209 "BER: Basic Encoding Rules for ASN.1"
ITU-T Recommendations, 1988

[14] PKCS#7 "Cryptographic Message Syntax Standard"
RSA Laboratories, Version 1.5, Nov 1993

[15] R. Housley “Cryptographic Message Syntax”
Internet Draft: draft-ietf-smime-cms-03.txt, Jan 1998

[16] Fielding et al “Hypetext transfer Protocol – HTTP/1.1”
Technical Report RFC 2068, Internet Engineering Task Force, Jan 1997

[17] D. Crocker “Augmented BNF for Syntax Specifications: ABNF”
Internet Engineering Task Force, Internet Draft, Oct 1996, Work In Progress

[18] J. C. Mogul and S. E. Deering “Path MTU Discovery”
Technical Report RFC 1191, Internet Engineering Task Force, Nov 1990

[19] W. R. Stevens “TCP/IP Illustrated: The Protocols”
Vol 1, Reading, Massachusetts. Pub: Addison-Wesley, 1994

[20] N. Freed and N. Borenstein “MIME: Part One. Format of internet Message Bodies”
Technical Report RFC 2045, Internet Engineering Task Force, Nov 1996

N. Freed and N. Borenstein “MIME: Part Two. Media Types”
Technical Report RFC 2046, Internet Engineering Task Force, Nov 1996

K. Moore “MIME: Part Three. Message header Extensions for Non-ASCII Text”
Technical Report RFC 2047, Internet Engineering Task Force, Nov 1996

N. Freed et al “MIME: Part Four. Registration Procedures”
Technical Report RFC 2048, Internet Engineering Task Force, Nov 1996

N. Freed and N. Borenstein “MIME: Part Five. Conformance Criteria and Examples”
Technical Report RFC 2049, Internet Engineering Task Force, Nov 1996

[21] Character sets assigned by IANA. See <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>

[22] ANSI X9.42

[23] ANSI X9.57-1997x, “Public Key Cryptography for the Financial Services Industry: Certificate Management”. Working Draft, June 1996

[24] ANSI X3.106 “American National Standards for Information Systems – Data Link Encryption”
American National Standards Institute, 1983 (DES)

W. Tuchman “Hellman Presents No Shortcuts Solutions to DES”
IEEE Spectrum, v16, n7, pp40-41, July 1979

[25] R.L. Rivest et al “Description of the RC2 Encryption Algorithm”
Internet Draft: draft-rivest-rc2desc-01.txt

[26] NIST FIPS PUB 180-1 “Secure Hash Standard”
National Institute of Standards and Technology, U.S. Dept. of Commerce, DRAFT, May 1994

[27] R. L. Rivest “The MD5 Message Digest Algorithm”
Technical Report RFC 1321, Internet Engineering Task Force, April 1992

[28] Bradner “Key words for use in RFCs to indicate Requirement Level”
Technical Report RFC 2119, Internet Engineering Task Force, March 1997

Authors’ Addresses

Peter Kirstein, Goli Montasser-Kohsari and Edmund Whelan are all at University College London. Their contact details are:

P.Kirstein@cs.ucl.ac.uk	Tel: +44 171 380 7286
G.Montasser-Kohsari@cs.ucl.ac.uk	Tel: +44 171 380 7215
E.Whelan@cs.ucl.ac.uk	Tel: +44 171 419 3688

Dept of Computer Science   Fax: +44 171 387 1397
University College London
Gower Street
London WC1E 6BT England

Acknowledgements

The European Commission, under the Telematics 1007 “MERCI” project and the Telematics 1005 “ICE-TEL” project funded the design of SIP Security. Mark Handely, Henning Schulzrinne and Eve Schooler developed the original SIP draft.