Internet Engineering Task Force MMUSIC WG
Internet Draft Peter Kirstein
draft-ietf-mmusic-sap-sec-04.txt> Goli Montasser-Kohsari
March 12th 1998 Edmund Whelan
Expires: September 12th 1998 University College London

SAP Security Using Public Key Algorithms

Status of this Memo

Abstract

The Session Announcement Protocol (SAP) has been specified in such a way that authentication and privacy can be assured. However the algorithms and mechanisms to achieve such security are not prescribed in the current draft. This document extends the SAP protocol, by describing specific algorithms and formats of authentication and encryption formats based on PGP and PKCS#7 standards. It is a companion document to draft-ietf-mmusic-sap.

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.

Glossary

ASN	Abstract Syntax Notation
CT	Content Type
CTB	Cipher Type Byte
DA	Digest Algorithm
DEA	Digest Encryption Algorithm
DES	Data Encryption Standards
EAID	Encryption Algorithm Identifier
EK	Encryption Key
EKID	Encryption Key Identifier
IETF	Internet Engineering Task Force
MD	Message Digest
MMUSIC	Multiparty Multimedia Session Control
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
PH	Privacy Header
PK	Public Key
PKCS	Public Key Cryptographic System
PKCS	Public Key Cryptography Standard (as in PKCS#7)
SAP	Session Announcement Protocol
SDP	Session Descriptor Protocol
SEK	Session Encryption Key
SK	Secret Key
SGK	Secret Group Key
SHA	Secure Hash Algorithm

Table of Contents

TABLE OF CONTENTS **3**

1. INTRODUCTION **4**

2. AUTHENTICATION AND ENCRYPTION OF ANNOUNCEMENTS **4**

2.1 INTRODUCTION 4
2.2 SYMMETRIC AND ASYMMETRIC ENCRYPTION 5
2.3 AUTHENTICATED ANNOUNCEMENTS 5
2.4 AUTHENTICATED AND ENCRYPTED ANNOUNCEMENTS 6
2.4.1 Introduction *6*
2.4.2 Distribution of Session Encryption Keys *6*
2.4.3 Use of Public Key Algorithms *7*
2.4.4 Encrypting Announcements *8*
2.4.5 Decrypting Announcements *8*

3. SECURED SAP PACKET FORMATS **9**

3.1 SECURED SAP PACKET FORMAT 9
3.2 AUTHENTICATION HEADER 11
3.2.1 Generic Format *11*
3.2.2 PGP Format *12*
3.2.3 PKCS#7 Format *12*
3.3 ENCRYPTED PAYLOAD FORMAT 13
3.3.1 Generic Format *13*
3.3.2 Symmetric Encryption *13*
3.3.3 Hybrid Encryption *14*
3.3.3.1 PGP Format Privacy Header 14
3.3.3.2 PKCS#7 Format Privacy Header 15
3.3.4 Supported Algorithms *15*
3.3.4.1 Symmetric Encryption 15
3.3.4.2 Hybrid Encryption 15
3.3.4.2.1 PGP Format 15
3.3.4.2.2 PKCS#7 Format 15

4 CHANGES FROM PREVIOUS DRAFT **16**

1. Introduction

An Mbone session directory is used to advertise multimedia conferences, and to communicate the session addresses (whether multicast or unicast) and conference-tool-specific information necessary for participation. Such sessions are announced using the Session Announcement Protocol (SAP) described in a companion draft [1]. The SAP protocol allows for the incorporation of authentication of the announcement originator, and for privacy of the session details; however neither the choice of authentication algorithms used, nor the mechanisms for encrypting the SDP Session Description, are detailed in the draft.

This document describes the format of the authentication header for SAP data packets that use security services based on PGP [2] or PKCS#7 [3]. The SAP document also provides for the confidentiality services required for the SDP payload [4], which describes the conference set-up parameters. This document describes how both symmetric and hybrid symmetric/public key encryption algorithms should be used to provide private announcements.

Much of this document is concerned with security considerations. This document is currently in the process of peer review and, until the process has been completed, should not be considered authoritative in this area.

2. Authentication and Encryption of Announcements

2.1 Introduction

It is necessary to provide authentication and integrity of the Session Announcement to ensure that only authorised persons modify Session Announcements and to provide facilities for announcing securely encrypted sessions - providing the relevant proposed conferees with the means to decrypt the data streams. The Session Announcements are made to announce to all potential conferees the existence of a conference. It has, however, another function - to try to minimise conflicts for Mbone resources by spreading out the number of simultaneous conferences. Thus there are a number of threats which we must try to address in the securing of the Session Announcement, and some constraints. These include the following:

- Authentication and Integrity of Session Announcement

  Here it is necessary to ensure that the Session Announcement comes from the person claimed, and is indeed an authorised announcement. Since subsequent announcements will modify caches of future conferences, it is possible otherwise to spoof an original announcement, and thereby cause a Denial of Service attack

- Confidentiality of Conference Details for Session Announcement

Here it is at least necessary to hide the details of the addresses and media formats used. In order to minimise schedule conflicts; it is desirable to keep the time of a conference known, even if all other details are concealed.

Three types of announcement are supported: ‘unsecured’, ‘authenticated’ and ‘authenticated and encrypted’. The ‘unsecured’ type is described in the SAP specification [1] and so only the latter two types are described below.

2.2 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[10]) 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 session key which is different each time. This session key is then encrypted and distributed using public key algorithms.

Two complete systems, which can achieve both authentication and confidentiality using particular PKCS systems, are PGP [2] and PKCS#7 [3]; similar mechanisms are used, but different encryption algorithms and formats are used. The differences between the algorithm and format details for these two systems are elaborated in Sections 3.2 and 3.3. As detailed later implementers MUST support PGP formats with support for PKCS#7 being OPTIONAL.

2.3 Authenticated Announcements

In order to send authenticated announcements it is possible to use the algorithms of either PGP [2] or the PKCS#7 [3] systems. The resulting format will be substantially different; the exact details are given in Sections 3.2 and 3.3. For each format, the announcement originator calculates a message digest of the announcement. The originator's secret key is used to encrypt the message digest, together with an electronic timestamp, thus forming a digital signature. The originator sends the digital signature along with the message; the receiver receives the message and the digital signature, and recovers the original message digest from the digital signature by decrypting it with the sender's public key. The receiver computes a new message digest from the message, and checks to see if it matches the one recovered from the digital signature. If it matches, then this is considered adequate proof that the message was not altered, and that it came from the originator who owns the public key used to check the signature.

Each Session announcement contains a message ID hash [4]. The specifications for SAP announcements [1] states that such announcements may be repeated frequently, but that if any change is made in the announcement, a different message ID must be used; as a result, a different message ID hash will be appended to the message. As a result, it is only necessary to authenticate an announcement the first time it is received.

To save space in the announcement message, only a public key identifier is generally included. It is then assumed that the public key itself has either been distributed previously or can be retrieved from a cache or directory. Optionally the Public Key itself can be included in the announcement in the form of a certificate removing the need for prior distribution. Consequently, providing that the Public Key is already available in a local cache or Directory, or is distributed with the announcement, one can be sure that the same originator sent the announcement. Only if the full public key information and a Certificate Authority infrastructure are accessible can the originator be identified [5].

2.4 Authenticated and Encrypted Announcements

2.4.1 Introduction

When it is desired to make private announcements, it is necessary to encrypt the set-up details of the conference. The normal way of providing such encryption is to use only a symmetric encryption algorithm such as the Data Encryption Standard (DES [6]) to encrypt such a session using a Session Encryption Key (SEK); this algorithm is used because other systems, such as the asymmetric RSA system [10], are too computation-intensive for large amounts of data - though they are economic for smaller amounts. For symmetric encryption systems, the SEK must be securely distributed to all authorised recipients.

2.4.2 Distribution of Session Encryption Keys

There are various ways that the SEK could be distributed; all rely on distributing some shared secret in advance to the intended participants in the conference group. When this process takes place out of band, it is not described further in this document.

Many symmetric encryption algorithms, e.g. DES [6] are known to be easy to break; with such algorithms, it is undesirable to re-use the SEK many times. For this reason, and to improve security, a set of SEKs may be distributed out-of-band; the recipients may then try to decrypt the announcement by trying each of these SEKs in turn.

As in Section 2.3, one may use the fact that if any change is made in the announcement, a different message ID, and hence message ID hash is used; it is only necessary to attempt to decrypt an announcement message the first time it is received. security, and ensuring the receipt, managing the cache, and trying several keys can be relatively complex and expensive.

The basic symmetric system is contained in SAP [1]. To improve efficiency, it would be possible to use symmetric encryption with a pre-distributed Key Identifier (KeyID). However, because of the potential weakening of the security by the use of KeyIDs, and the consequent need to use more secure symmetric algorithms, we do not recommend this technique. Moreover, by adopting the use of asymmetric Public Key technology for such SEK distribution as discussed below, we gain both efficiency and have a better integrated approach to authentication.from the PH. When this mechanism is used, because the SEK inside the PH is encrypted using a strong encryption algorithm, the same Encryption Key may be re-used - thus reducing the number of shared secrets which have to be distributed to the intended recipients. There is now no objection to using an Encryption Key Identifier to refer to the stronger encryption key of the Public Key system.

2.4.3 Use of Public Key Algorithms

Public Key Cryptography is one mechanism for minimising the need for secure transmission of shared secrets; this was already used in the Authenticated Announcements of Section 2.2. It would be possible to use these encryption algorithms on the whole announcement message but this would be inefficient because the asymmetric encryption algorithms normally use much longer encryption keys, and are much more resource-intensive, than the symmetric ones. For this reason it is more efficient to use a combination of symmetric and Public Key algorithms. Now a random Session Encryption Key (SEK) is generated as in Section 2.4.1. A Privacy Header (PH) is constructed containing the SEK, which is encrypted with the asymmetric encryption algorithm. It is now only necessary to distribute a Secret Group Key (SGK) and Public Group Key (PGK) i.e. {SGK, PGK} to decrypt the SEK. However, this pair needs to be distributed only once as long as the group membership does not change; it is possible to re-use the same group keys for many sessions, with different SEKs. This minimises the number of times the prior key distribution sequence must be followed.

It should be noted that because a Group Key is used in the above, it is not possible to use the same Header to authenticate the sender uniquely, though it is authenticated automatically that the sender is one of the group which has reserved the asymmetric encryption key pair. It is still possible to authenticate the identity of the sender by using a different Authentication Key for the Authentication Header as described in Section 2.3.

It would be possible to use a similar technique using symmetric encryption with a strong encryption algorithm and an encryption key Identifier instead of the Public Key Group Key, However, we believe the Public Key method to be superior so this variant is not pursued.

2.4.4 Encrypting Announcements

We will now provide some more detail. If the payload is to be compressed, this is performed prior to encryption as the probability that ciphertext can be appreciably compressed is small.

When an announcement is to be encrypted, the payload is encrypted using symmetric encryption. In this case each such encryption key is used only once; a new Session Encryption Key (SEK) is generated as a random number for each announcement. Since it is to be used only once, the SEK is bound to the message and transmitted with it in a Privacy Header. The sequence is as follows: The Privacy Header contains the SEK, encrypted with the group's Public Group Key, together with information identifying the Group Key which has been used. The encrypted Payload is appended to the Privacy Header.

2.4.5 Decrypting Announcements

Upon receiving a new announcement with the encryption bit set, a receiver should attempt to decrypt the announcement with the relevant group private key or their own private key as indicated in the Privacy Header. The sequence is as follows:

1. Prior to the announcement, the group's Public/Secret Group Key pair is distributed securely.

2. From the announcement, the receiving participants determine which Public Group Key has been used by looking at the information contained in the Privacy Header.

3. They then decrypt the Session Encryption Key (SEK) with the SGK corresponding to the PGK identified in Step 2 and obtain other necessary information such as the content encryption algorithm from the Privacy Header.

4. The authorised receivers decrypt the encrypted text payload using the SEK and the relevant symmetric content encryption algorithm, which was used to encrypt the payload.

Note that if an encrypted announcement is being announced via a proxy, then there may be no way for the proxy to discover that the announcement has been superseded, and so it may continue to relay the old announcement in addition to the new announcement. SAP provides no mechanism to chain modified encrypted announcements, so it is advisable to announce the unmodified session as deleted for a short time after the modification has occurred. This does not guarantee that all proxies have deleted the session, and so receivers of encrypted sessions should be prepared to discard old versions of session announcements that they may receive (as identified by the SDP version field). In most cases however, the only stateful proxy will be local to (and known to) the sender, and an additional (local-area) protocol involving a handshake for such session modifications can be used to avoid this problem.

3. Secured SAP Packet Formats

Both Authentication and Privacy can be achieved using PGP [2] or PKCS#7 [3] format packets. Implementers MUST support PGP format with support for PKCS#7 being OPTIONAL. In Section 3.1 we discuss the generic packet format defined in SAP [1]. In Section 3.2 we consider the formats of the Authentication Header, and in Section 3.3 that of the encrypted payload.

It would be possible to define our own versions of the packets for this application. In that case the formats would be simpler, but all the implementations would have to be coded using the basic encryption libraries, and a new infrastructure would have to be defined. Both PGP and PKCS#7 already have complete implementations and, by using their formats, several application tool kits are already available (e.g. Entrust [14], Secude [15]). In addition, these formats also have complete infrastructures defined around them. For these reasons, we have chosen to retain enough compatibility to ease the eventual implementation, while simplifying the formats as far as possible within such a constraint. There is an additional advantage in this approach; it will be possible to send session announcements by the encrypted Session Invitation Protocol or by electronic mail using PGP or S-MIME, and re-use much of the same code as with SAP.

3.1 Secured SAP Packet Format

The SAP data packets as defined in [1] has the following format:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=2 |  MT |E|C|  Header Length |      16 bit  Message ID Hash  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Originating  Source                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Authentication Header (Optional)              |
|                            . . . . . . . .                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32 bit Timeout Field     (Optional)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Privacy Header          (Optional)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Text Payload (possibly encrypted)             |
|                            . . . . . . . .                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Notes:

Version Number, V: This is a 3-bit field and has the value 2 for this version of SAP [1]

Message Type, MT: This defines the contents of the payload and can be

0 Session Descriptor Announcement Packet in which case the payload is an SDP session description as described in [4]

1 Session Description Deletion Packet in which case the payload is a single SDP line containing the origin field of the announcement to be deleted

Encryption Bit, E: -. If this is set, the text payload has been encrypted

Compression Bit, C: If this is set the payload has been compressed using the gzip compression utility [7]

Header length: This is an 8 bit unsigned quantity giving the number of 32 bit words following the main SAP header that contains the authentication data. If this is non-zero, the payload is authenticated, and an Authentication Header is present

Message Identifier Hash: This is a 16 bit quantity that, when used in combination with the originating source, provides a globally unique id identifying the precise version of this announcement. The message id hash should be changed if any field of the session description is changed. A value of zero means that the hash should be ignored and the message should always be parsed.

Originating Source: This 32-bit field gives the IP address of the original source of the message. It is permissible for this to be zero if the message has not passed through a proxy relay and if the message id hash is also zero. This is intended for backward compatibility with SAPv0 clients only.

Authentication Header: This can be used for two purposes:

1 Verification that changes to a session description or deletion of a session are permitted

2 Authentication of the identity of the session creator.

In some circumstances only verification is possible because a certificate signed by a mutually trusted person or authority is not available. However, under such circumstances, the session originator can still be authenticated to be the same as the session originator of previous sessions claiming to be from the same person. This may or may not be sufficient, depending on the purpose of the session and the people involved. The precise format of the Authentication Header is discussed in Section 3.2.

Timeout: When the session payload is encrypted, and the session description is being relayed or announced via a proxy, the detailed timing fields in the SDP description are not available to the proxy. This is because these fields are encrypted and the proxy is not trusted with the decryption key. Under such circumstances, SAP includes an additional 32-bit timestamp field stating when the session should be timed out. The digital signature in the authentication header encompasses the time-out so that a session cannot be maliciously deleted by modifying its time-out in an announcing proxy. The value is an unsigned quantity giving the NTP time [8] in seconds at which time the session is timed out. It is in network byte order and MUST be present when encryption has been used.

Privacy Header: This is present when the text payload has been encrypted using hybrid encryption.

Text Payload: When there is no encryption, the encryption bit is not set and this format is as defined in the SDP [4] draft. However, when encryption has been used the payload is encrypted and the format is discussed in Section 3.3.

3.2 Authentication Header

3.2.1 Generic Format

The generic format of the Authentication Header is given below. The structure of the Format Specific Authentication Subheader, using both the PGP and the PKCS#7 formats, is discussed in Sections 3.2.2 and 3.2.3 respectively.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=1 | P|  Auth  |Signature Length|                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                             |
|           Format Specific Authentication Subheader            |
|                           . . . . . . . .                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Notes:

Version Number, V: For this release the version number is 1 (3 bits)

Padding Bit, P: If necessary the data in the Authentication Header is padded to be a multiple of 32 bits and the Padding bit is set. In this case the last byte of the Authentication Header contains the number of padding bytes (including the last byte) that must be discarded.

Authentication Type, Auth: The Authentication Type is a 4 bit encoded field that denotes the authentication infrastructure the sender expects the recipients to use to check the authenticity and integrity of the information. This defines the format of the Authentication Subheader and can be:

1 PGP Format
2 PKCS#7 Format
3 PGP Format with the ‘Certificate’ included.

4 PKCS#7 with the Certificate include

Signature Length: This gives the length of the Signature. This is necessary for the case when the Authentication Type is 3 to allow the beginning of the PGP Certificate Packet to be found. In the PKCS#7 cases (when the Authentication Type is 2 or 4) this is unnecessary and SHOULD be set to zero.

3.2.2 PGP Format

Implementations MUST support this format. The generic description of the PGP packets is presented in [2]. For PGP the basic Format Specific Authentication Subheader comprises a digital signature packet as described in [2]. This involves the use of a hash code, or message digest algorithm, and a public key encryption algorithm. The hash is taken of the text payload together with the signature classification (1 byte) and signature time stamp (4 bytes) fields as described in [2]. For the case when the Authentication type is 1 the Subheader contains a Digital Signature Packet only with the hexadecimal signature classification being <00> or <01>. The only Message Digest Algorithm is 1 (MD5) and the Public Key Cryptosystem (PKC) is 1, this being the RSA system. If the type is 3 then a Certificate Packet is also appended to the previous Signature Packet. A certificate packet is composed of the following individual packets:

(a) A Public Key Packet which defines the RSA public key
(b) A UserID packet
(c) A Signature Packet

The Public Key packet again has the Public Key Cryptosystem 1. In the case of Signature Packet (c) the signature classification now takes the hexadecimal values <10> to <13>. These packets are all detailed in [2].

3.2.3 PKCS#7 Format

Support for this format is OPTIONAL. The Format Specific Authentication Subheader will, in the PKCS#7 case, have an ASN.1 ContentInfo type with the ContentType being signedData. Use is made of the option available in PKCS#7 to leave the content itself blank as the content which is signed is, in this application, already present as the text payload, whether this is encrypted or not. Thus inclusion of it within the SignedData type would be duplication and increase the packet length unnecessarily. The full specification for this ASN.1 type is available in [3].

There will only be one signerInfo and related fields corresponding to the originator of the SAP announcement. Although it would be possible to transfer the relevant information is a single signerInfo type rather than the complete ContentInfo it is considered preferable to use the latter for two reasons. Firstly, this is compatible with a wider range of applications and security toolkits and secondly, that the certificate can be included in a standard way. If the Authentication Type is 2 there are no certificates or certificate revocation lists whereas if the authentication type is 4 the originator's X.509 certificate is added in the certificate field of this type.

In addition, for both type 2 and 4, use is made of the ability in PKCS#7 to authenticate various attributes as specified in PKCS#9 [16]. The authenticatedAttributes field of the SignerInfo type is used, the authenticated attribute being the SigningTime. This is a mandatory field in this specification. Consequently, the initial input to the message digesting process is the contents octets of the DER encoding of the content field of the ContentInfo value (not including the identifier octets or the length octets). Moreover, the signing time is an authenticated attribute. Thus, the result of the message digest is the complete DER encoding of the Attributes value contained in the AuthenticatedAttributes field. This contains the SigningTime in addition to the content type and message digest of the payload, which are included automatically.

3.3 Encrypted Payload Format

3.3.1 Generic Format

The format of the Encrypted Payload depends on the type of encryption used to encrypt the SDP text payload [4]. If no encryption has been used only the Text payload is present.

If encryption has been used then the encryption bit in the main SAP header is set and the payload is encrypted either symmetrically or using hybrid encryption. For symmetric encryption the format is detailed in Section 3.3.2 whereas hybrid encryption is detailed in Section 3.3.3. For hybrid encryption there are two possibilities - PGP and PKCS#7 Formats. The application is expected to test whether the fields immediately following the timeout field in the main SAP header are compatible with the use of symmetric encryption; in this case it will be a padding bit followed by a 31-bit random field, or whether it is compatible with the use of hybrid encryption. In this case there is a very specific format to the first byte of the Privacy header, which follows other time-out field in this case.

3.3.2 Symmetric Encryption

If symmetric encryption alone has been used then the encrypted payload has a random field added prior to encryption as below.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| P|                  31 bit Random Field                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Text Payload                          |
|                          . . . . . . . .                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Notes:

Random Field: This field is only present when payload is encrypted using symmetric encryption and is used to perform the randomisation tasks normally performed by an initialisation vector in algorithms such as DES. This 31-bit field should contain a genuinely random number.

Padding Bit, P: This bit indicates that the payload was padded prior to encryption. The last byte of the encrypted payload indicates how many padding bytes were added.

The data following the Time-out field is decrypted using the algorithm specified above. Further details on the encryption algorithms are given in [6, 12, 13].

3.3.3 Hybrid Encryption

If a combination of asymmetric and symmetric encryption has been used then the part of the SAP packet following the time-out field has the following structure. This effectively takes the form of a Privacy header followed by the encrypted SDP payload, the precise format depending on whether PGP or PKCS#7 formats have been used. The specific details for each of these formats are described in Sections 3.3.3.1 and 3.3.3.2 respectively and, as with the authentication, implementations MUST support PGP with PKCS#7 being OPTIONAL.

Privacy Header:

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=1 |P| Type  |  Header Length  |                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                             |
|                Format Specific Privacy Header                 |
|                           . . . . . . . .                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Notes:

Version, V: In this version the Version of the privacy Header is 1

Padding Bit, P: If necessary the data in the Privacy Header is padded to be a multiple of 32 bits and the Padding bit is set. In this case the last byte of the Privacy Header contains the number of padding bytes (including the last byte) that must be discarded.

Format Type, Type: This can be either 1 for PGP or 2 for PKCS#7 format.

Header Length: This gives the length of the Privacy Header.

3.3.3.1 PGP Format Privacy Header

For the case when the Format Type is 1 the Format Specific Privacy Subheader is composed of a PGP Public Key encrypted packet and the text payload is a PGP Conventional Key Encrypted Packet. These are detailed in [2]. The Public Key Cryptosystem is 1, this being defined as the RSA system, and the only supported symmetric encryption algorithm is the IDEA algorithm, corresponding to the Conventional encryption type byte value of 1.

3.3.3.2 PKCS#7 Format Privacy Header

If the Type is 2 then the Format Specific Privacy Header is composed of a PKCS#7 ContentInfo type with the ContentType being envelopedData. These are detailed in [2]. In this case the Text Payload, which has been symmetrically encrypted with the algorithm specified in the contentEncryptionAlgorithm field, is effectively the encryptedContent part of the structure. There will be only one recipientInfo structure corresponding to the group certificate, which has been previously distributed. The contentType in the EncryptedContentInfo structure is "Data".

3.3.4 Supported Algorithms

3.3.4.1 Symmetric Encryption

If symmetric encryption alone is used then DES [6] MUST be supported.

3.3.4.2 Hybrid Encryption

3.3.4.2.1 PGP Format

- *Content Encryption* – the IDEA symmetric encryption algorithm with a key length of 128 bits MUST be supported.
- *Digest Algorithm* – the MD5 [18] Message Digest Algorithm MUST be supported.
- *Digest Encryption Algorithm* – the asymmetric rsaEncryption algorithm [10] MUST be supported with key sizes from 512 bits to 1024 bits.
- *Key encryption Algorithm* - the asymmetric rsaEncryption algorithm [10] MUST be supported with key sizes from 512 bits to 1024 bits.

3.3.4.2.2 PKCS#7 Format

In order to maintain wide interoperability the algorithms supported here follow [23] where fuller details can be found.

- **KeyEncryptionAlgorithmIdentifier**

  Sending and receiving agents MUST support Diffie-Hellman defined in [21].

  Receiving agents SHOULD support rsaEncryption [10]. 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.

- **SignatureAlgorithmIdentifier**

  Sending and receiving agents MUST support id-dsa defined in [22]. The algorithm parameters MUST be absent (not encoded as NULL).

  Receiving agents SHOULD support rsaEncryption, defined in [10]. 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.

- **ContentEncryptionAlgorithmIdentifier**

  Receiving agents MUST support encryption and decryption with DES EDE3 CBC [17]. Receiving agents SHOULD support encryption and decryption using the RC2 [20] or a compatible algorithm at a key size of 40 bits.

- **DigestAlgorithmIdentifier**

  Receiving agents MUST support SHA-1 [19]. Receiving agents SHOULD support MD5 [18]. Sending agents SHOULD use SHA-1.

4 Changes from Previous Draft

Section 2.2 rewritten. Other minor changes in text. A table of contents has been added and a "Signature Length" field has been added to the Authentication Header to allow access to the PGP certificate when the Authentication type is 3. Support for PGP is now specified as MANDATORY with support for PKCS#7 being OPTIONAL.

References

[1] M.Handley ‘SAP: Session Announcement Protocol” INTERNET-DRAFT, draft-ietf-mmusic-sap-00.txt, 11/27/1996.

[2] D.Atkins, ”PGP Message Exchange Formats”, RFC 1991, August 1996.

[3] PKCS#7, Cryptographic Message Syntax Standard, RSA Laboratories, Version 1.5, November 1993

[4] M.Handley, V. Jacobson, ‘‘SDP: Session Description Protocol”, INTERNET-DRAFT, draft-ietf-mmusic-sdp-02.txt, 11/27/1996.

[5] R. Housley , W. Ford , T. Polk Internet Public Key Infrastructure INTERNET- DRAFT, draft-ietf-pkix-ipki-part1-03.txt December 1996.

[6] National Bureau of Standards, Data Encryption Standard, Federal Information Processing Standards Publication 46, January 1977

[7] P. Deutsch, ‘‘GZIP file format specification version 4.3”, RFC 1952, May 1996.

[8] D. Mills, ‘‘Network Time Protocol version 2 specification and implementation", RFC1119, 1st Sept 1989.

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

[10] PKCS #1 RSA Encryption Standard RSA Laboratories, Version 1.5, November 1993

[11] H. Schulzrinne, ‘‘RTP Profile for Audio and Video Conferences with Minimal Control” RFC 1890, January 1996

[12] P. Metzger, P. Karn, W. Simpson, The ESP Information Triple DES-CBC Transformation, 10/02/1995 RFC850

[13] ANSI X3.92-1981. American National Standards Data Encryption Algorithm. American National Standards Institute, Approved 30 December 19990

[14] For details of ENTRUST see http://www.entrust.com/

[15] For details of SECUDE see http://www.darmstadt.gmd.de/secude/

[16] PKCS#9 Selected Attribute Types, RSA Laboratories, Version 1.1, November 1993

[17] W. Tuchman, "Hellman Presents No Shortcut Solutions to DES"
IEEE Spectrum, v. 16, n. 7, July 1979, pp 40-41

[18] "The MD5 Message Digest Algorithm" RFC 1321

[19] NIST FIPS PUB 180-1 "Secure Hash Standard" National Institute of Standards and Technology, U.S. Department of Commerce, DRAFT, 31

[20] "Description of the RC2 Encryption Algorithm", Internet Draft draft-rivest-rc2desc.

[21] ANSI X9.42

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

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

Authors' Addresses

Peter Kirstein, Goli Montasser Kohsari and Edmund Whelan are at University College London and 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

SAP and SDP were originally based on the protocol used by the sd session directory from Van Jacobson at LBNL. The European Commission under the Esprit 7602 "MICE" project, the Telematics 1007 "MERCI" project and the Telematics 1005 "ICE-TEL" project funded the design of SAP and SAP Security.