SIP Security

Andreas Steffen, Daniel Kaufmann und Andreas Stricker

Security Group
Zürcher Hochschule Winterthur
CH-8401 Winterthur
andreas.steffen@zhwin.ch

Abstract: Ubiquitous worldwide broadband Internet access as well the coming of age of VoIP technology have made Voice-over-IP an increasingly attractive and useful network application. Currently the "human-readable" Session Initiation Protocol (SIP) which is based on a simple HTTP-like request/response exchange is steadily gaining headway against the considerably more complex ASN.1 encoded H.323 Multimedia ITU-T standard introduced by the telecom industry some years ago. Unfortunately little attention has been given to the security aspects involved in running a phone connection over the public Internet. This paper gives a comparative overview over the security mechanisms recommended by the SIP standard and presents a practical SIP implementation realized at the Zürcher Hochschule Winterthur (ZHW), based on S/MIME authentication and encryption of the session initiation and ensuing protection of the media channels using the Secure Real-time Transport Protocol (SRTP).

1 The Session Inititation Protocol (SIP)

Due to its simple and fast session setup mechanism, the Session Initiation Protocol (SIP) [Ro02] has quickly made large inroads into the Voice-over-IP (VoIP) market previously dominated by implementations adhering to the rather complex H.323 ITU-T Internet telephony standard. Whereas H.323 is closely modelling a traditional ISDN Layer 3 call setup and uses ASN.1-coded binary messages for signalling, SIP is based on an HTTP-like request/response transaction model using human-readable ASCII messages with a syntax nearly identical to HTTP/1.1 [Fi99]. Figure 10 depicts an example of a SIP INVITE request which includes all necessary information required to set up an audio connection.

1.1 Example SIP Session

Figure 1 shows a typical SIP message exchange scenario between two users Alice and Bob belonging to the domains *atlanta.com* and *biloxi.com*, respectively. SIP user identification is based on a special type of Uniform Resource Identifier (URI) called a SIP URI with a form similar to an email address. In our example Alice's SIP URI is assumed to be *sip:alice@atlanta.com* and Bob's *sip:bob@biloxi.com*.

In order to establish a multimedia connection over the Internet between Alice's and Bob's *User Agents* (UAs) which can be either hardware SIP phones or PC based softphones, Bob's SIP URI must first be resolved into the IP address of the UA under which Bob is currently registered. SIP address resolution and routing is usually not done by the UA

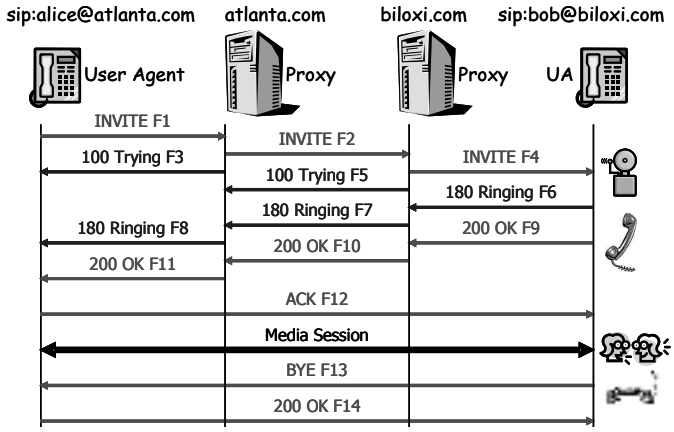

Figure 1: Session Initiation between two User Agents

itself but delegated to the *proxy server* responsible for the domain the UA is attached to. In our example the *atlanta.com* proxy will make a DNS lookup to determine the proxy server of the *biloxi.com* domain on behalf of Alice's user agent. The SIP INVITE request originating from Alice's UA is then forwarded via the *atlanta.com proxy* to the *biloxi.com* proxy which with the help of a location service determines the current whereabouts of Bob's user agent. Both the informational *Ringing* message and the *OK* message which is issued when Bob accepts the call, take the return path via the proxy server hops whereas the *ACK* message and the payload packets of the ensuing multimedia session will use the direct path between the two user agents.

1.2 The SIP Trapezoid

Thus the typical message flow during a SIP session takes on the form of a trapezoid as shown in Figure 2. From the point of view of network security this means that both the individual hops must be secured on a hop-by-hop basis as well as the direct path between the user agents. SIP session management messages are usually embedded into UDP datagrams but can also be transported over a TCP stream if the SIP message size comes within the physical medium's Maximum Transmission Unit (MTU) or if the underlying security mechanism requires a TCP connection. On the other hand the Real-time Transport Protocol (RTP) [Sch03] employed in media sessions exclusively uses non-reliable UDP datagrams to transport real-time audio and video packets over the Internet.

This means that any security mechanism employed to encrypt and authenticate multimedia streams must support UDP as a transport protocol. This requirement excludes certain security popular solutions like e.g. TCP based *Transport Layer Security* (TLS) [DA99]. The following chapter will give an overview on the choice of security mechanisms that can be selected to ensure data integrity and confidentiality for both the SIP based session management and the real-time transmission of multimedia payloads.

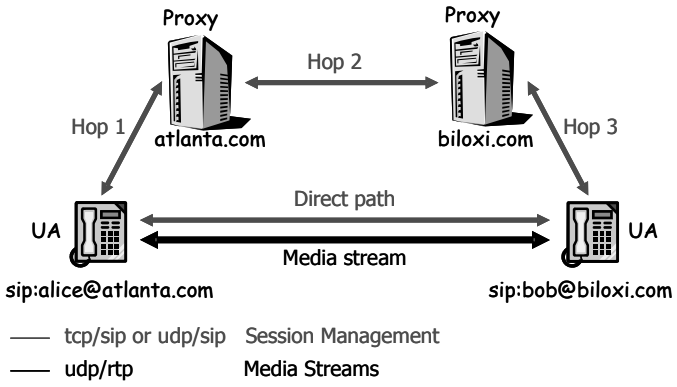

Figure 2: Basic SIP trapezoid

2 Security Mechanisms

2.1 Securing the SIP Session Management

Since the SIP message structure is a straight derivation from the HTTP request/response model, all security mechanisms available for HTTP [Fr99] can also be applied to SIP sessions. On the other hand the use of MIME containers within SIP messages suggests the potential use of email security mechanisms like PGP [El96] or S/MIME [Ra99]. And of course similar to a *https:* URI, a corresponding *sips:* URI will try to build up a secure transport layer tunnel using TLS [DA99]. And last but not least IP security (IPsec) [KA98] can be used as a general purpose mechanism to encrypt all IP based communication right on the network layer.

The major security mechanisms suited for the protection of a SIP session are shown in Figure 3. They coincide with the list of methods recommended by version 1 of the SIP standard [Ha99]. In the meantime two of the methods, namely HTTP basic authentication and PGP have been deprecated by version 2 of the Session Initiation Protocol [Ro02]. The pros and cons of the individual approaches were examined in detail by [KS03]. In this paper we will just give a concise summary of the findings.

HTTP Basic Authentication

HTTP basic authentication [Fr99] requires the transmission of a username and a matching password embedded in the header of a HTTP request. Included in a SIP request this user information could be used by a SIP proxy server or destination user agent to authenticate a SIP client or the previous SIP hop in a proxy chain. Because the cleartext password can be easily sniffed and therefore poses a serious security risk, the use of HTTP basic authentication has been deprecated by SIPv2.

HTTP Digest Authentication

HTTP digest authentication [Fr99] improves on the deficiencies of the HTTP basic authentication approach by transmitting an MD5 or SHA-1 *digest* of both the secret password and

Authentication methods: PSK Pre-Shared Keys PKI Public Key Infrastructure	Authentication	Data Integrity	Confidentiality	
HTTP 1.0 Basic Authentication	PSK	-	-	Deprecated by SIPv2 Insecure transmission of password
HTTP 1.1 Digest Authentication	PSK	-	-	Challenge/response exchange based on MD5 hash of [strong] password
Pretty Good Privacy (PGP)	PKI	✓	✓	Deprecated by SIPv2
Secure MIME (S/MIME)	PKI	✓	✓	For encryption the public key of the recipient user agent must be known
SIPS URI (TLS)	PKI	✓	✓	SIP application and proxies must tightly integrate TLS
IP Security (IPsec)	PKI	✓	✓	Integration with SIP application not required but proxies must be trusted

Figure 3: Solutions for securing the SIP session management

a random challenge string in place of the vulnerable password itself. More details on the digest challenge/response protocol plus some example messages can be found in chapter 4. Although HTTP digest authentication has the advantage that the identity of the user can be established without the need to transmit passwords in the clear, the procedure can still become easy prey to off-line dictionary attacks based on intercepted hash values if short or weak passwords are used. Another big disadvantage is the lack of an encryption mechanism to ensure confidentiality. Neither can the integrity of the SIP messages be guaranteed.

Pretty Good Privacy (PGP)

Pretty Good Privacy [El96] could be potentially used to authenticate and optionally encrypt MIME payloads contained in SIP messages but version 2 of SIP has deprecated the use of PGP in favour of S/MIME.

Secure MIME (S/MIME)

SIP messages carry MIME bodies and the MIME standard includes mechanisms for securing MIME contents to ensure both integrity and confidentiality by means of the *multipart/signed* and *application/pkcs7-mime* MIME types, see [Ga95, Ho99, Ra99]. X.509 certificates are used to identify the end users on the basis of their email addresses which are part of the SIP URI (in our example *alice@atlanta.com* and *bob@biloxi.com*, respectively). The signing of MIME bodies is the lesser problem since each user is in possession of her private key and the user certificate may be forwarded to the recipient embedded into the *pkcs7-mime* or *pkcs7-signature* attachments [Ka98]. On the other hand the encryption of MIME bodies, e.g. the Session Description Protocol (SDP) payload [HJ98] requires the foreknowledge of the recipient's public key. This key, usually certified by X.509 certificate must either be pre-fetched from a public directory or may be requested from the peer via a special SIP message. S/MIME based protection is treated in more depth in chapter 5.

Any mechanisms depending on the existence of end-user certificates are seriously limited in that there is virtually no consolidated authority today that provides certificates for end-user applications on a global scale. Because self-signed certificates are prone to man-in-the-middle attacks, either certificates from known public certification authorities (CAs) should be used or private CAs must be mutually recognized.

SIPS URI (TLS)

The use of a SIPS URI of the form *sips:bob.biloxy.com* in an INVITE message requires that TLS must be used on the whole path to the destination. Since each hop may add route information to the SIP message header, TLS protection must be realized on a hop-by-hop basis on each segment of the path. The use of TLS requires the use of TCP as a transport protocol (tcp/sip) and necessitates a public key infrastructure.

IP Security (IPsec)

IPsec is a general purpose mechanism that can be used to protect the SIP messages right on the network level. Due to the requirement that each proxy server on the path must have read/write access to the SIP header, IPsec ESP (Encapsulating Security Payload) or AH (Authentication Header) in transport mode [KA98] must be applied on a hop-by-hop basis. The necessary IPsec security associations can either be established on a permanent basis without active involvement of the SIP user agents using the connections or might be set up on the fly by the UAs and proxy servers themselves by tight interaction with the underlying IPsec stack.

The Internet Key Exchange (IKE) protocol [HC98] which is used to set up IPsec security associations supports both Pre-Shared Key (PSK) and Public Key (PKI) based authentication. Because the IP addresses of the SIP user agents will be mostly dynamic and taking into account that IKE Main Mode in that case does not work with pre-shared secrets and that IKE Aggressive Mode is fraught with security problems (man-in-the-middle attacks, off-line dictionary attacks on the PSK, etc.), public key based authentication will be the preferred method. This means that the establishment of global trust into X.509 certificates becomes again the major problem.

2.2 Securing the Real-time Media Streams

Multimedia streams are packet-oriented and are transported using the unreliable UDP based Real-time Transport Protocol (RTP) [Sch03]. In order to have some feedback about the quality of the received packet stream, the peer periodically sends back a report using the companion Real-time Transport Control Protocol (RTCP). Since live audio and video connections are very sensitive to both absolute time delay and large delay variations (jitter), any packet encryption and authentication algorithm should not significantly influence these parameters. Due to these real-time restraints and the prerequisite that the transmission must be UDP based, only the two security mechanisms listed in Figure 4 are currently available.

Secure RTP (SRTP)

The Secure Real-time Transport Protocol (SRTP) [Ba04] is an extension to the RTP Audio/Video profile [SC03] and provides confidentiality, message authentication, and replay

Authentication methods: PSK Pre-Shared Keys PKI Public Key Infrastructure	Authentication	Data Integrity	Confidentiality	
Secure RTP (SRTP)	PSK	✓	✓	Uses master key which must be distributed by other means
IP Security (IPsec)	PKI	✓	✓	Integration with SIP application not required but peer must be trusted

Figure 4: Solutions for securing the real-time media streams

protection to the RTP and RTCP traffic. The use of the cryptographically powerful but computationally efficient AES cipher running in stream cipher mode guarantees strong security but does not increase the size of the encrypted payload. The authentication tag required for data integrity adds 10 bytes to each RTP/RTCP packet but could be weakened to 4 bytes if a very narrow-band communications channel is being used. More in-depth information on SRTP will be presented in the next chapter.

IP Security (IPsec)

As an alternative the real-time payloads can be protected right on the network layer by IPsec transport mode, using the same security association already negotiated to protect the SIP message exchange between the user agents. A serious drawback might be the large overhead incurred by the ESP encapsulation which in the worst case amounts to 37 bytes per IPsec packet for 3DES encryption (8 bytes ESP header, 8 bytes IV, 2-9 bytes ESP trailer, and 12 bytes HMAC) and up to 53 bytes for AES encryption (8 bytes ESP header, 16 bytes AES IV, 2-17 bytes ESP trailer, and 12 bytes HMAC). E.g. if an ITU-T G.711 A-law or μ-law audio codec is used which generates an 8 bit speech sample every 125 μs and 10 ms of uncompressed speech is mapped as 80 contiguous samples into a single RTP packet then the IPsec overhead is still between 30-50 %.

3 The Secure Real-Time Transport Protocol (SRTP)

Both RTP and RTCP packets can be cryptographically secured by the Secure Real-time Transport Protocol (SRTP) and the companion Secure Real-time Transport Control Protocol (SRTCP), respectively [Ba04]. This chapter gives the details on the message formats and sheds some light on session key generation and master key distribution.

3.1 The Secure RTP Packet Format

The SRTP message format is shown in Figure 5. As can be seen, only the RTP payload body (including any RTP padding if present) is encrypted. Since all currently defined encryption transforms do not add any padding, the size of the RTP payload is not increased by encryption.

The Master Key Identifier (MKI) field is optional and identifies the master key from which the session keys were derived. The MKI can be used by the receiver to retrieve the correct master key when the need for a re-keying event comes up.

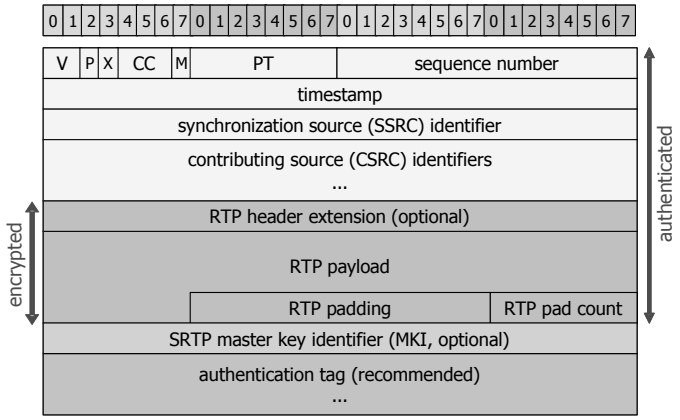

Figure 5: Secure RTP packet format

The 16 bit sequence number already present in the RTP packet is used together with a 32 bit rollover counter (ROC) which is part of the cryptographic context for the SRTP session to prevent replay attacks.

The authentication tag is a cryptographic checksum computed over both the header and body of the RTP packet. Its use is strongly recommended since it protects the packets from unauthorized modification. The default tag length is 10 bytes but might be reduced if the transmission channel does not allow such a large increase of the RTP packet size.

3.2 The Secure RTCP Packet Format

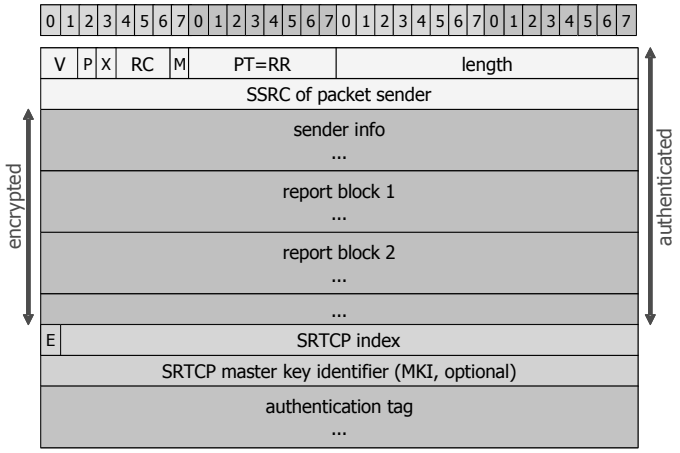

Figure 6: Secure RTCP packet format

Figure 6 shows that the RTP control packets are secured in a similar way as the RTP packets themselves, one difference being that the use of the authentication tag is mandatory. Otherwise it would be possible for a malevolent attacker e.g. to terminate an RTP media stream by sending a BYE packet. An additional field is the SRTCP index which used as a sequence counter preventing replay-attacks. The MSB of the index field is used as an Encryption flag (E) which is set if the RTCP body is encrypted.

3.3 Default Encryption Algorithms

In principle any encryption scheme can be used with SRTP. As default algorithms the NULL cipher (no confidentiality) and the Advanced Encryption Standard in Counter Mode (AES-CTR) are defined. The AES-CTR encryption setup is shown in Figure 7.

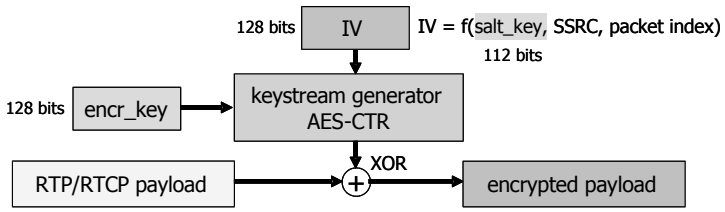

Figure 7: Encryption using AES in counter mode.

AES in counter mode acts as a *keystream generator* producing a pseudo-random keystream of arbitrary length that is applied in a bit-wise fashion to the RTP/RTCP payload by means of a logical XOR function, thus working as a classical stream cipher. AES itself is a block cipher with a block size of 128 bits and a key size of 128, 192, or 256 bits. In order to work as a pseudo-random generator AES is loaded at the start of each RTP/RTCP packet with a distinct initialisation vector (IV) that is derived by hashing a 112 bit salt_key, the synchronisation source identifier (SSRC) of the media stream, and the packet index. Encrypting this IV results in an output of 128 pseudo-random bits. Next the IV is incremented by one and again encrypted, thus generating the next 128 bits of the keystream. By counting the IV up by increments of one as many keystream blocks can be generated as are required to encrypt the whole RTP/RTPC payload. Any remaining bits from the last keystream block are simply discarded.

AES used in counter mode instead of the more common cipher block chaining mode (CBC) has the big advantage that the keystream can be precomputed before the payload becomes available thus minimizing the delay introduced by encryption. And of course by using a stream cipher instead of block cipher there is no need to pad the payload up to a multiple of the block size which would add 15 overhead bytes to the RTP/RTCP packet in the worst case.

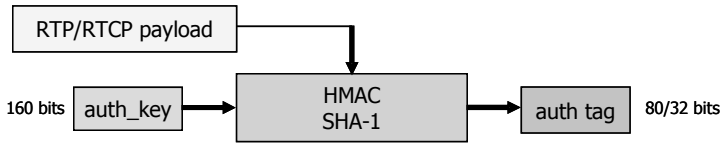

Figure 8: Authentication using HMAC-SHA-1

3.4 Default Authentication Algorithm

The default SRTP message authentication algorithm is HMAC-SHA-1 [KBC97], based on the popular 160 bit SHA-1 hash function. Cryptographical security is achieved by hashing a 160 bit secret *auth_key* into the checksum which is then truncated to 80 bits in order to reduce the packet overhead and which has the further advantage that it hides the complete internal state of the hash function. In applications where transmission bandwidth is a problem the authentication tag might be weakened to 32 bits.

3.5 Session Key Derivation

The encryption and authentication algorithms described in sections 3.3 and 3.4 both require secret symmetric session keys that must be known to all user agents participating in a SIP session. This raises the logistical problem of session key generation and distribution. The SRTP standard offers a partial solution by deriving all needed session keys from a common master key but leaves open the distribution of the master key itself.

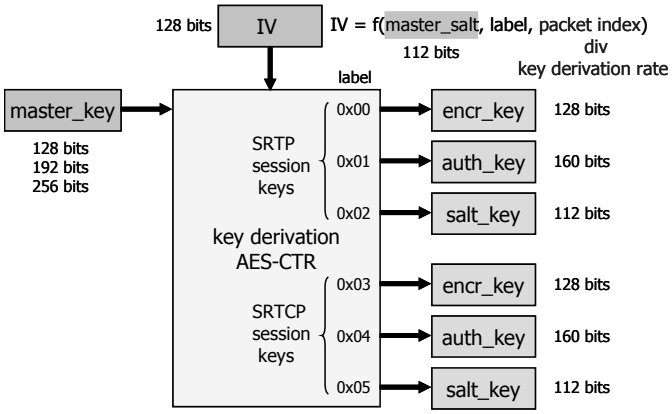

Figure 9: Session key derivation

Figure 9 shows how the session keys are computed starting out from a single master key. Again the AES block cipher is used in counter mode to generate the necessary keying material. The master key which can have a size of 128, 192, or 256 bits plays the role of

the AES encryption key. The pseudo-random generator is loaded with an IV that is itself a function of a 112 bit *master_salt* value, a one byte *label* and a *session key number* . By applying the labels *0x00* up to *0x05*, encryption, authentication and salting keys for both SRTP and SRTCP are derived from the same master key. If a *key derivation rate* has been defined then every time a number of packets equivalent to the key derivation rate have been sent, a new set of either SRTP or SRTCP session keys are computed. If the key derivation rate is set to zero then the same set of keys is used for the whole duration of the session.

3.6 Master Key Distribution

We turn now to the crucial issue of distributing the master key to the user agents as part of the session initiation. An approach proposed by [Ba04] is to use *Multimedia Internet KEYing* (MIKEY) [Ar03] to establish the cryptographic SRTP context. MIKEY is a new general key exchange protocol similar to IPsec's Internet Key Exchange (IKE) [HC98] but tailored to the specific demands of a multimedia environment. Unfortunately, since no reference implementation is readily available, the use of MIKEY is not an option yet.

As a workaround the *k* key parameter defined by the Session Description Protocol (SDP) [HJ98] could be used to transfer the master key. Figure 10 shows a typical SIP INVITE message that carries all parameters needed to set up a multimedia session embedded in an *application/sdp* MIME body.

```
INVITE sip:bob@zhwin.ch SIP/2.0
Via: SIP/2.0/UDP 160.85.170.139:5060;branch=z9hG4bK4129d28b8904
To: Bob <sip:bob@zhwin.ch>
From: Alice <sip:alice@zhwin.ch>;tag=daa21162
Call-ID: 392c3f2b568e92a8eb37d448886edd1a@160.85.170.139
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:alice@dskt6816.zhwin.ch:5060>
Content-Type: application/sdp
Content-Length: 239

v=0
o=alice 3157331353 3157331353 IN IP4 160.85.170.139
s=DA SIP Security 2003
c=IN IP4 160.85.170.139
t=0 0
k=clear:910bc4defa71eb6190008762fca6ae2f1d959e87cdf3c0c5c5076ad38ee8
m=audio 10000 RTP/AVP 0
a=ptime:20
a=rtpmap:0 PCMU/8000
```

Figure 10: SIP INVITE request carrying an SDP MIME body

In the example shown above the *k* parameter defines an 128 bit SRTP master key in hexadecimal notation. Of course a cleartext transmission of the master key would pose a severe security risk. Under the assumption that the proxy servers can be trusted the complete SIP INVITE could be encrypted on a hop-by-hop basis using either TLS or IPsec.

If end-to-end confidentiality of the SDP MIME body is desired then S/MIME protection as described in chapter 5 should be used as an alternative.

4 HTTP Digest Authentication

This chapter shows how a SIP INVITE request originating from the alleged user Alice is authenticated by the first hop proxy server using HTTP Digest Authentication [Fr99].

```
SIP/2.0 407 Proxy Authentication Required
Via: SIP/2.0/UDP 160.85.170.139:5060;branch=z9hG4bK4129d28b8904
To: Bob <sip:bob@zhwin.ch>;tag=3b6c2a3f
From: Alice <sip:alice@zhwin.ch>;tag=daa21162
Call-ID: 392c3f2b568e92a8eb37d448886edd1a@160.85.170.139
CSeq: 1 INVITE
Proxy-Authenticate: Digest algorithm=MD5,
nonce="1058800787",
realm="zhwin.ch"
Content-Length: 0
```

Figure 11: HTTP digest authentication – challenge

The proxy server receiving the SIP INVITE message from Alice immediately replies with the challenge shown in Figure 11. The challenge contains a random nonce and defines the digest algorithm to be used (usually MD5 or SHA-1) as well as the realm for which the user must provide an authentication.

Upon reception of the challenge Alice resends the original INVITE request but inserts the response to the challenge into the SIP message header as shown in Figure 12. The response value consists of the MD5 digest of the username, the secret password, the nonce value, the SIP method and the requested URI. Thus the password is not transmitted in the clear but must be known by the proxy server in order to be able to verify the authentication response.

5 Secure MIME (S/MIME)

S/MIME [Ra99] can be used to secure MIME bodies either on a hop-by-hop basis or end-to-end from user agent to user agent. Figure 13 shows how the SDP MIME attachment embedded in the SIP INVITE request of Figure 10 can be encrypted and signed using S/MIME. The *application/pkcs7-mime* binary *envelopedData* structure encapsulates the symmetrically encrypted SDP payload and also contains the symmetric key which is encrypted with the public key of the recipient. In our example the encrypted payload is additionally signed using the *multipart/signed* method [Ga95]. The signature plus optionally the X.509 certificate of the signer is contained in the binary *application/ pkcs7-signature* structure which is attached after the MIME object to be signed. As an alternative a binary PKCS#7 *signedData* structure [Ka98] could be used which transports both the data to be signed and the signature within a single attachment.

```
INVITE sip:bob@zhwin.ch SIP/2.0
Via: SIP/2.0/UDP 160.85.170.139:5060;branch=z9hG4bK4129d28b8904
To: Bob <sip:bob@zhwin.ch>
From: Alice <sip:alice@zhwin.ch>;tag=daa21162
Call-ID: 392c3f2b568e92a8eb37d448886edd1a@160.85.170.139
CSeq: 2 INVITE
Proxy-Authorization: Digest algorithm=MD5,
nonce="1058800787",
realm="zhwin.ch",
response="142311a910a4d57ba49afdbe5646768c",
uri="sip:bob@zhwin.ch",
username="alice"
Max-Forwards: 70
Contact: <sip:alice@dskt6816.zhwin.ch:5060>
Content-Type: application/sdp
Content-Length: 239
<Session Description Protocol not shown>
```

Figure 12: HTTP digest authentication – authenticated INVITE request

```
INVITE sip:bob@zhwin.ch SIP/2.0
Via: SIP/2.0/UDP 160.85.170.139:5060;branch=z9hG4bK4129d28b8904
To: Bob <sip:bob@zhwin.ch>
From: Alice <sip:alice@zhwin.ch>;tag=daa21162
Call-ID: 392c3f2b568e92a8eb37d448886edd1a@160.85.170.139
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:alice@dskt6816.zhwin.ch:5060>
Content-Type: multipart/signed;boundary=992d915fef419824;
micalg=sha1;protocol=application/pkcs7-signature
Content-Length: 3088
--992d915fef419824
Content-Type: application/pkcs7-mime;
smime-type=envelopeddata; name=smime.p7m
Content-Disposition: attachment;handling=required;filename=smime.p7m
Content-Transfer-Encoding: binary
<envelopedData object encapsulating encrypted SDP attachment not shown>
--992d915fef419824
Content-Type: application/pkcs7-signature;name=smime.p7s
Content-Disposition: attachment;handling=required;filename=smime.p7s
Content-Transfer-Encoding: binary
<signedData object containing signature not shown>
--992d915fef419824--
```

Figure 13: S/MIME encrypted and authenticated SDP MIME attachment

6 Practical Results

Three students of the Zurich University of Applied Sciences in Winterthur, Switzerland wrote a simple *SIPsec* user agent for the Linux operating system as part of their diploma thesis [GLS03]. They based their client on the *reSIProcate* SIPv2 stack available from http://www.resiprocate.org which already integrates some basic S/MIME support by making use of the popular OpenSSL cryptographic library. They added C++ code of their own which automatically generates a random SRTP master key and transmits it embedded in an encrypted SDP MIME body. The X.509 certificates required for the S/MIME encryption and authentication operations were created with *TinyCA* from *tinyca.sm-zone.net*.This

Public Key Infrastructure (PKI) tool presents itself with a comfortable GUI that hides the rather scary OpenSSL command line interface.

For the secure transport of RTP media streams and the corresponding RTCP control streams the *libsrtp* library from *srtp.sourceforge.net* was used. The current release of the library supports 128 bit AES counter mode encryption and HMAC-SHA-1 authentication but does not yet implement re-keying by defining a key derivation rate different from zero, nor does it support the inclusion of the Master Key Index (MKI) field in SRTP or SRCTP messages.

7 Conclusions

The diploma thesis [GLS03] showed as a *proof-of-concept* that S/MIME encryption and authentication of SDP MIME attachment works and can be used to securely transfer a SRTP master key which can then be used to protect the RTP media streams. Therefore because of the possibility of end-to-end encryption the use of S/MIME in SIP messages is an attractive alternative to the hop-by-hop security offered by TLS.

At the current time, similar to S/MIME protected email, the establishment of trust into peer certificates on a global scale remains one of the open problems yet to be solved.

References

[Ar03] Arkko, J. et.al.: MIKEY: Multimedia Internet KEYing. IETF Internet Draft <draft-ietf-msec-mikey-08.txt>, 2003.

[Ba04] Baugher, M. et.al.: The Secure Real-Time Transport Protocol (SRTP). IETF RFC 3711, 2004.

[DA99] Dierks, T.; Allen, C.: The TLS Protocol Version 1.0, IETF RFC 2246, 1999.

[El96] Elkins, M.: MIME Security with Pretty Good Privacy (PGP), IETF RFC 2015, 1996

[Fi99] Fielding, R. et al.: Hypertext Transfer Protocol - HTTP/1.1, IETF RFC 2616, 1999.

[Fr99] Franks, J. et. al.: HTTP authentication: Basic and Digest Access Authentication", IETF RFC 2617, 1999.

[Ga95] Galvin, J. et al.: Security Multiparts for MIME: Multipart/Signed and Multipart/ Encrypted, IETF RFC 1847, 1995.

[GLS03] Gisler, A.; Loretz. M.; Stricker, A.: SIP Security, Diplomarbeit, Zürcher Hochschule Winterthur, 2003.

[Ha99] Handley, M. et. al.: SIP : Session Initiation Protocol Version 1, IETF RFC 2543, 1999.

[HC98] Harkins, D.; Carrel, D.: The Internet Key Exchange (IKE), IETF RFC 2409, 1998

[HJ98] Handley, M.; Jacobson, V.: SDP: Session Description Protocol, IETF RFC 2327, 1998.

[Ho99] Housley, R.: Cryptographic Message Syntax, IETF RFC 2630, 1999.

[KA98] Kent, S.; Atkinson, R.: Security Architecture for the Internet Protocol, IETF RFC 2401, 1998.

[Ka98] Kaliski, B.: PKCS #7: Cryptographic Message Syntax Version 1.5, IETF RFC 2315, 1998.

[KBC97] Krawczyk, H.; Bellare, M,; Canetti, R.; HMAC: Key-Hashing for Message Authentication, IETF RFC 2104, 1997.

[KS03] Kaufmann, D.; Stricker, A.: SIP Security, Projektarbeit, Zürcher Hochschule Winterthur, 2003.

[Ra99] Ramsdell B.: S/MIME Version 3 Message Specification, IETF RFC 2633, 1999.

[Ro02] Rosenberg, J. et.al.: SIP: Session Initiation Protocol Version 2. IETF RFC 3261, 2002.

[Sch03] Schulzrinne, H. et.al.: RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550, 2003.

[SC03] Schulzrinne, H.; Casner, S.: RTP Profile for Audio and Video Conferences with Minimal Control. IETF RFC 3551, 2003.