KTH Microelectronics
and Information Technology

Alternatives to MIKEY/SRTP to secure VoIP

Master of Science Thesis

JOACHIM ORRBLAD

Stockholm/Kista March 2005

Telecommunication System Laboratory
KTH Microelectronics and Information Technology

Preface

This work was conducted as a master thesis project at the Telecommunication Systems Laboratory (TSLab) of the department of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (KTH), Stockholm/Kista between September 2004 and March 2005.

Examiner:	Professor Björn Pehrson	<bjorn@imit.kth.se>
Supervisors:	Jon-Olov Vatn	<vatn@imit.kth.se>
	Erik Eliasson	<eliasson@imit.kth.se>
	Johan Bilien	<bilien@imit.kth.se>

I would like to express my sincere gratitude to these wonderful people who have guided me through this thesis project with their expertise, feedback and most of all there patience with me and my ideas.

Abstract

Security for Voice over IP (VoIP) can be achieved in different ways and can be divided into two main aspects. Securing the *call signaling* i.e. the IP traffic used for establishing the call and securing the *call itself*, here referred to as the media session. This thesis focus on the security for the media session although the two aspects are strongly related.

KTH has released an open source Session Initiation Protocol (SIP) user agent to demonstrate VoIP functionality. This agent currently uses Secure Real-Time Protocol (SRTP) to secure the media session and Multimedia Internet KEYing (MIKEY) for exchanging keying materials for SRTP. This thesis will examine IP security (IPSEC) as an alternative to MIKEY/SRTP and ways to integrate the key exchange for IPSEC in the SIP call signaling.

My conclusion in this thesis is that SRTP should be used to secure VoIP, but SIP initiated IPSEC makes it possible to establish IPSEC tunnels between persons who do not know each others IP addresses before the call. General IPSEC tunnels can be used to protect all traffic between these two persons, not only the VoIP call.

The chosen and implemented solution, for the key exchange, is based on SIP, MIME and MIKEY. Linux native IPSEC support is used for encryption and authentication.

Table of Contents

1 Introduction

The goal of this thesis is to find an alternative to MIKEY/SRTP for a Secure VoIP media session. There are basically three parts of this goal, where part one concerns the alternatives for MIKEY/SRTP, and the second part is to implement the chosen solution into *minisip*. Part three is to evaluate that implementation and compare it with alternate ways of securing the media session, e.g MIKEY/SRTP.

The focus of this thesis is on key exchange for IPSEC in the context of VoIP, i.e. alternatives to manual keying and *Internet Key Exchange* (IKE) [RFC 2401] being able to be integrated with SIP. IKE is used to negotiate IPSEC security parameters between two *hosts*. IP telephony is usually established between two *persons*, thus IKE cannot be used right away, since the caller does not generally know the IP address of the callee's host. For encryption and authentication of the media stream, existing IPSEC solutions will be used.

The goal of the thesis can be explained in more detail as follows:

- Establish an IPSEC connection to secure the audio streams between two hosts running *minisip*.
- The IPSEC connection should be initiated by the minisip user agent (UA).
- Keep the number of round trips needed for the keying mechanism as low as possible.
- The keying mechanism should, if possible, use SIP signaling as transport to reduce round trips.
- Find a keying mechanism that fits the requirements mentioned above.
- The IPSEC connection should be able to protect general traffic, not only the traffic generated by the media.

The choice of IPSEC as the alternative to evaluate was not primarily done by evaluating different alternatives, instead it was chosen mainly on its qualification of being a well known concept amongst many and the fact that it applies security on the network layer in contrast to SRTP that applies the security on the application layer. IPSEC can also be used to protect general IP traffic not only the VoIP call. This makes SIP initiated IPSEC a way of establishing general VPN tunnels between endpoints defined by their users.

The outline of this thesis report is as follows. In section 2 the involved technologies are described. In section 3 existing solutions relating to this thesis are presented. Section 4 and 5 contain possible solutions and the chosen solution. Section 6 contains the measurements done in this thesis and section 7 the thesis conclusions. Future work and references in section 8 and 9.

2 Technologies involved

Minisip uses open protocol standards to set up and maintain VoIP sessions. The most important protocols and technologies are in this chapter given a short presentation.

2.1 SIP (Session Initiation Protocol)

To establish a VoIP session between two persons/hosts signaling is needed to find each party of the call. SIP [RFC 3261] is such a signaling protocol.

2.1.1 General

When trying to make a VoIP call the caller needs to find the callee. In the world of the *circuit switched telephony* each subscriber has an unique telephone number identifying the telephone of the subscriber. The relationship between the subscriber e.g. John Doe and his telephone number is more or less static, and usually found in the phone book. When a call is made a connection is established between the well known locations of the caller and the callee.

IP telephony uses IP addresses to find the way from the caller to the callee. The connection between John Doe and his IP address is loose and may change over time. For instance the IP telephony client might have got its IP address dynamically from DHCP server and may differ each time the client connects. If the caller do not know the IP address of the callee, SIP is a way to find out and establish the connection. If the caller already knows the IP address of the callee the session parameters could be negotiated directly by other means than SIP, but SIP can of course still be used as a convenient and consistent way of establishing the connection.
The identity of a caller and a callee is a kind of URI (Universal Resource Identifier) [RFC 3986] and have a syntax similar to the one of an e-mail address e.g. SIP:john@doedomain.org, except for the prefix SIP:, and is called the SIP identity or SIP URI [RFC 3261].
The SIP URI is used by the caller to find the callee. The IP address corresponding to the SIP URI is found in the proxy server that the URI is associated with. That proxy is found in a way similar to the finding of an e-mail server corresponding to an e-mail address through use of the DNS *SRV* records [RFC 3263] instead of *MX* records. SIP is not only used for locating the IP address of a SIP URI but also for negotiating session parameters of the media streams e.g. codecs so that the session can be established.

One of the great benefits of SIP is that both *finding* the callee and *negotiating* of session parameters can be done within the same protocol. VoIP supports a number of media types and SIP uses SDP (Session Description Protocol) [RFC 2327] to communicate supported ones.

2.1.2 SIP architecture

There and five entities in SIP, registrar, location, proxy, UA (User Agent) and redirect server. The UA (User Agent) is the phone and the registrar receives registrations and requests updates of the location server, which keep track of the UA's. The UAC (User Agent Client) and the UAS (User Agent Server) are the UA that makes the call (caller) and the UA that receives the call (callee). Each registrar belongs to a domain in a similar way as a mail server can belong to a domain. The proxy routes SIP messages on behalf of the UA. Redirect servers directs UA's to alternate URI. Usually the registrar, location and the proxy runs on the same server.

The UA registers with its registrar when it goes on line to tell the registrar that it is available. This is done with the SIP message REGISTER, see figure below.

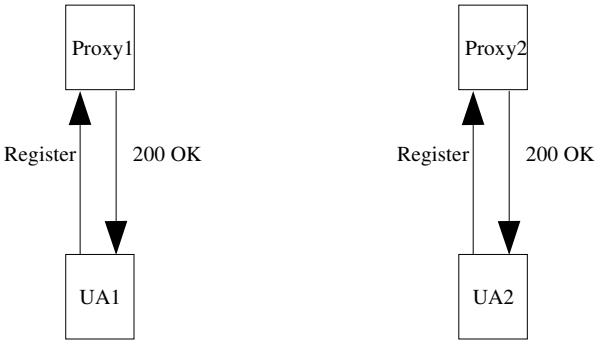


fig.1

The REGISTER message contains information on how the UA can be reached, thus when the UA is registered in the registrar the UA can be reach by other UA's.

An example of a REGISTER message is shown below.

REGISTER sip:registrar.doedomain.org SIP/2.0
To: <sip:john@doedomain.org>
From: <sip:john@doedomain.org>; tag=randomunique#
Call-ID: random#@doedomain.org
Cseq: 4711 REGISTER
Contact: <sip:john@johnspc.doedomain.org:5060>;expires=1000
Max-Forwards: 70
Via: SIP/2.0/UDP johnspc.doedomain.org:5060;branch=z9hG4bKrandomunique#

expires: 7200
Content-Length: 0

2.1.3 Making a call

A standard call can be described with the SIP trapezoid when UA1 wants to establish a call to UA2, where the following SIP messages are involved:

- INVITE: used by the caller to initiate a call to the callee
- Trying: a response from the next-hop server that the INVITE message is received and processed. Used by many UA but not mandatory.
- Ringing: alerting the caller that the callee has received the INVITE
- 200 OK: the request has succeeded e.g. call answered or hang up
- ACK: establish the media session
- BYE: hang up call, This signal may be sent via the proxies.

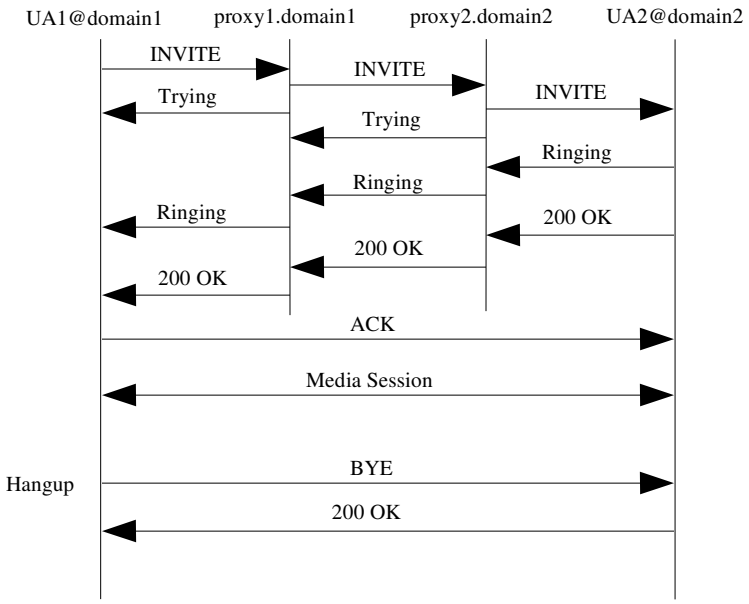


fig.2

Preferred and supported session parameters from the caller are encapsulated in the INVITE message in a SDP body, and the parameters chosen by the callee are encapsulated in the 200 OK message. The message 12 ACK, 13 BYE, 14 200 OK, may go via the proxy. The DNS lookups that are needed in figure 2 are not shown.

Example of an INVITE message from caller John to callee Alice below

```
INVITE sip: alice@callee.org SIP/2.0
To: <sip:alice@callee.org>
From: <sip:john@doedomain.org>; tag=randomunique#
Call-ID: unique#@doedomain.org
Cseq: 4711 INVITE
Contact: <sip:john@johnspc.doedomain.org:5060;user=phone;transport=UDP>
Max-Forwards: 70
Via: SIP/2.0/UDP johnspc.doedomain.org:5060;branch=z9hG4bKrandomunique#
Content-Type: application/sdp
Content-Length: 176
(sdp body not shown)
```

The fields in the SIP body are as follows:

To: logical recipient of a request.
From: logical identity of the initiator of the request.
Call-ID: an unique identifier to group a series of messages. It must be the same for all requests and responses sent by either UA in a dialog.
Cseq: identification and order of transactions.
Contact: contains the the URI at which the UA would like to receive requests.
Max-Forwards: limits the number of hops a request can transit.
Via: indicates the transport used for the transaction and identifies the location where the response is to be sent.
Content-Type: indicates the media type of the message-body sent to the recipient.
Content-Length: size of message body.

2.2 SDP (Session Description Protocol)

2.2.1 General

SDP is defined in RFC 2327 [RFC 2327]

To establish multimedia sessions over the Internet one needs to have a way of describing the attributes of the session. SDP is such a protocol. If you want to establish a multimedia session you can announce the description of the session, through SDP, which may include:

- Session name and purpose
- Time the session is active
- The type of media (video, audio, etc)
- The transport protocol (RTP/UDP/IP, H.320, etc)
- The format of the media (H.261 video, MPEG video, etc)
- Information about where to receive those media (addresses, ports, etc.)

2.2.2 SDP used by SIP

How SDP is used by SIP is defined in RFC3264 [RFC3264]

To establish a VoIP call the caller need to negotiate session parameters with the callee. A call can consist of several multimedia channels e.g. voice and video etc., where each channel needs a unique set of parameters that describes the session. This negotiation can be done with SIP. The caller sends a SDP body that is encapsulated in the SIP INVITE message, with proposed parameters. The callee responds with a SDP body in the OK message with the the chosen parameters, so in just one round trip a common pair is negotiated.

Example of an INVITE with SDP body below

```
INVITE sip: alice@callee.org SIP/2.0
To: <sip:alice@callee.org>
From: <sip:john@doedomain.org>; tag=randomunique#
Call-ID: unique#@doedomain.org
Cseq: 4711 INVITE
Contact: <sip:john@johnspc.doedomain.org:5060;user=phone;transport=UDP>
Max-Forwards: 70
Via: SIP/2.0/UDP johnspc.doedomain.org:5060;branch=z9hG4bKrandomunique#
Content-Type: application/sdp
Content-Length: 139

v=0
o= 123 123 IN IP4 johnspc.doedomain.org
s=Minisip session
c=IN IP4 johnspc.doedomain.org
t=0 0
m=audio 32869 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1
```

The fields in the SDP body are as follows:

v =(protocol version) but also marks the beginning of the session description
o= (owner/creator and session identifier)
s= (session name)
c= (connection information - not required if included in all media)
t= (time the session is active)
m= (media name and transport address) but also marks the beginning of the media description
a= (media attribute lines)

2.3 IPSEC (Internet Protocol Security)

2.3.1 General

IPSEC [RFC 2401] [netsec] was designed to add security at the network layer i.e. adding security to all protocols above the network layer. IPSEC may make use of three protocols where the first two are for data protection:

- ESP [RFC2406], Encapsulating Security Payload - Encrypts and/or authenticates data.
- AH [RFC2402], Authentication Header - Provides a packet authentication service.
- IKE [RFC2409], Internet Key Exchange - Negotiates connection parameters, including keys, for the other two.

IPSEC as a term can be a bit indistinct since sometimes it includes all three protocols band sometimes only a subset. E.g., if there is a manual key exchange there is no need for IKE, and if the authentication that ESP provides is sufficient there is no need for AH.

IPSEC is intended to protect traffic between hosts and is applied at the network layer and can as such not supply the same kind of end to end security as protocols working at higher levels. Higher level security protocols can provide application to application security but IPSEC provides host to host security. It might have some implications on multiuser systems but in most cases it would not. Application to application is when a application, in this case *minisip,* does all encryption and decryption and do not rely on any other application for that service. In the case of IPSEC the kernel handle all security. Which traffic to protect with IPSEC is decided with the IPSEC policy. see section 2.3.3.

2.3.2 SA (Security Associations)

Each IPSEC secured connection is defined by a Security Associations (SA) [netsec] which contain secret keys, algorithms and IP addresses involved in the communication. The SA is considered unidirectional, so two SA are needed for bidirectional traffic. A SA contain the following information:

- Source and destination IP address of the resulting IPSEC header. These are the IP addresses of the IPSEC peers protecting the packets.
- IPSEC protocol (AH or ESP)
- The algorithm and secret key used by the IPSEC protocol.
- Security Parameter Index (SPI). This is a 32 bit number which identifies the security association.

and may contain:

- IPSEC mode (tunnel or transport, see section 2.3.5)
- Size of the sliding window to protect against replay attacks.
- Lifetime of the security association.

The different SA's are kept in a Security Association Database and is used when sending and receiving packets to retrieve adequate information to process the packets.

Example of SA

```
10.10.10.10 192.168.1.1
        esp mode=transport spi=11084(0x00002b4c) reqid=0(0x00000000)
        E: 3des-cbc  ea662e99 d71f3800 20e33276 e943a763 911dd75e bdf54974
        A: hmac-sha1  1c48be69 bd92c3a9 d1422c6a 4208b2d9
        seq=0x00000000 replay=64 flags=0x00000000 state=mature
        created: Feb 25 17:13:51 2005   current: Feb 25 17:14:22 2005
        diff: 31(s)     hard: 1073741824(s)     soft: 0(s)
        last: Feb 25 17:13:51 2005      hard: 1073741824(s)     soft: 0(s)
        current: 353576(bytes)  hard: 1073741824(bytes) soft: 0(bytes)
        allocated: 1525 hard: 200000000 soft: 31150981
        sadb_seq=1 pid=30397 refcnt=0
```

The above SA states that traffic from host 10.10.10.10 should be encrypted with 3des-cbc using the encryption key: ea662e99 d71f3... and authenticated with hmac-sha1 using the authentication key: c48be69 bd92c...

2.3.3 IPSEC policy

IPSEC requires a Security Policy Database containing the IPSEC policies [netsec] specifying which type of action to take for a specific packet. E.g. drop, protect or send in clear text. Decisions can be made on different fields in the packet e.g. source address, destination address, UDP or TCP. The IPSEC Security Policy is actually a filter that decides which traffic to protect.

Example of policy

```
10.10.10.10[any] 192.168.1.1[80] tcp
        out ipsec
        esp/transport//require
        created: Feb 25 17:13:51 2005  lastused: Feb 25 17:14:27 2005
        lifetime: 0(s) validtime: 0(s)
        spid=1905 seq=0 pid=30398
        refcnt=4
```

The above policy states that traffic from host 10.10.10.10 with any source port to host 192.168.1.1 and destination port 80 and transport protocol tcp must be protected with IPSEC.

2.3.4 IKE (Internet Key Exchange)

Internet Key Exchange (IKE) [RFC2409] [netsec]

The IKE protocol is used for setting up IPSEC (ESP/AH) connections between two hosts/gateways. IKE negotiation has two phases. Phase one is used for proving each other's identity and set up a secure connection (ISAKMP SA or IKE SA) for phase 2. Phase 2 is used for negotiating IPSEC SA (ESP/AH) so that the secure data connection can be established. Since a SA is unidirectional they are negotiated in pairs to handle two way traffic. The IKE SA can be used to negotiate more than one IPSEC SA. If there exist a IPSEC security policy that states that the traffic should be protected but there is no matching SA, IKE will try to negotiate a SA according to IKE configuration. If the negotiation fails and no new SA is created the traffic will be dropt. If the negotiation succeeds a new SA will be created.

Phase 1 can be either of two modes:
1. Main mode

Parameter negotiation
message 1: Alice >-- crypto suites supported --> Bob
message 2: Alice <-- chosen crypto suites --< Bob

Diffie-Hellman exchange
message 3: Alice >-- g^a mod p --> Bob
message 4: Alice <-- g^b mod p --< Bob

Send IDs and authenticate, encrypted
message 5: Alice >-- g^ab mod p {"Alice", proof I'm Alice} --> Bob
message 6: Alice <-- g^ab mod p {"Bob", proof I'm Bob} --< Bob

2. Aggressive mode

message 1: Alice >-- g^a mod p, "Alice", crypto proposal --> Bob
message 2: Alice <-- g^b mod p, "crypto choice, proof I'm Bob --< Bob
message 3: Alice >-- proof I'm Alice --> Bob

The main differences between mode 1 and 2 are, except for the number of packets, that mode 1 can negotiate the Diffie-Hellman (DH) number since the DH exchange is done in message 3&4 and that in aggressive mode sends Alice and Bob's identities unprotected.

Phase 2 (also known as "Quick mode")

Phase 2 IKE is a 3 message protocol that negotiates parameters for the phase 2 SA (ESP/AH SA), including cryptographic parameters and the SPI to identify the SA.

message 1: Alice >-- X, Y, K_{enc} {CP, traffic, SPI_A, $nonce_A$, [g^a mod p]} --> Bob
message 2: Alice <-- X, Y, K_{enc} {CPA, traffic, SPI_B, $nonce_B$, [g^b mod p]} <-- Bob
message 3: Alice >-- X, Y, ack --> Bob

where:

- X, is the cookie pair generated in phase 1
- Y, 23-bit number chosen by the phase 2 initiator to distinguish this phase 2 session from others within the same phase 1 session.
- CP, Crypto Proposal for SA
- CPA, Crypto Proposal Accepted
- traffic, optional description of the traffic to be sent.
- [g^x mod p] optional DH values.

The IKE protocol is currently being revised and IKEv2 is being developed and is defined in the INTERNET-DRAFT [ikev2] which expires, in its current version, in March 2005 and will obsolete RFC 2409 if adopted. The main objective with IKEv2 and the main difference from IKEv1 is simplification. [PK01]

2.3.5 IPSEC mode

IPSEC can be used in either of two modes, tunnel mode or transport mode. [netsec] In transport mode the IPSEC information (AH and/or ESP) is inserted between the IP header and the rest of the packet while in tunnel mode the original packet is intact and a new IP header and IPSEC information is added outside. Why two modes? Transport mode is used directly between two hosts and tunnel mode when establishing IPSEC tunnels between e.g. two firewalls when creating a VPN.

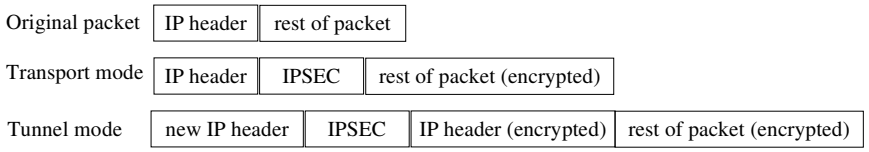


fig.3

2.3.6 ESP (Encapsulated Security Payload)

ESP can be used for encryption and integrity protection, ESP always uses encryption but integrity is optional. [RFC2406] [netsec] ESP is a rather odd header since it really encapsulates the encrypted data i.e. adding information both in front of the encrypted data and after. ESP can be used as integrity protection only if the special "null encryption" algorithm is used. The ESP header looks as follows:

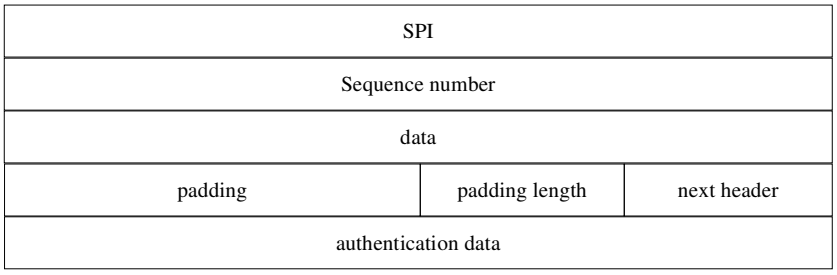


fig.4

- SPI (Security parameter index, identifies SA) [4 octets]
- sequence number (used for protection against replay attacks, this has nothing to do with TCP sequence number) [4 octets]
- IV (initialization vector is used by some cryptographic algorithms. The length of the field depends on the algorithm and once the SA is established the field length is fixed) [variable]
- data (This is the protected data) [variable] This is an encrypted field.
- padding (to make sure that the data is the correct size for the cryptographic algorithm and ensure that the combination of the fields data, padding, padding length and next header are a multiple of four octets.) [variable] This is an encrypted field.
- padding length (Number of octets of padding) [1 octet] This is an encrypted field.
- next header/protocol type (same as protocol field in IPv4 or Next header in IPv6) [1 octet] This is an encrypted field.
- authentication data (Cryptographic integrity check. The length is determined by the authentication function selected for the SA, if zero length ESP is providing encryption only) [variable]

2.3.7 AH (Authentication Header)

AH [RFC2402] [netsec] provides authentication only, if one wants encryption one must use ESP. Another difference between the ESP authentication and AH authentication is that AH also provides some protection of the IP header except for IP header field that can be modified by routers. For IPv4 AH the mutable fields are: Type of service, flags, fragment offset, TTL and header checksum.

The AH header looks as follows:

- next header (as ESP) [1 octet]
- payload length (The size of the AH header in 32-bit chunks, not counting the first 8 octets) [1octet]
- unused [2 octets]
- SPI (as ESP) [4 octets]
- sequence number (as ESP) [4 octets]
- authentication data (The cryptographic integrity check on the data) [variable]

2.4 MIKEY (Multimedia Internet KEYing)

Multimedia Internet KEYing MIKEY [RFC 3830] was designed to meet the requirements of initiation of secure multimedia sessions. That is:

- the parameters for the security protocol should be exchanged in one round trip.
- the protocol should be simple and straight forward.
- the protocol should be possible to transport in session establishment protocols e.g. SDP.
- the protocol should supply end-to-end security for the keying material.
- independence from any specific security functionality of the underlying transport.
- low bandwidth consumption and low computational workload.

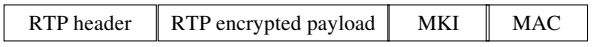
MIKEY supports three types of key agreements

1. Pre-shared key (PSK) - In this method the pre-shared secret is used to derive keys both for encryption and integrity of the MIKEY message. In the MIKEY message the random generated TGK (Traffic-encrypting key Generation Key) is securely transported. This is the most cost effective key agreement but the problem of distributing the shared secrets makes this solution hard to scale.
2. Public-key encryption (PKE) - Similar to PSK but the initiator makes a random key used for encryption and integrity. This key is then encrypted with the responders public key and sent to the responder. This approach is more resource consuming but has the ability to scale if a Public Key Infrastructure (PKI) is available.
3. Diffie-Hellman (DH) - The only method that supplies perfect forward secrecy. This approach is resource consuming and can only be used to establish single peer-to-peer keys. It also requires the existence of a PKI for message signing.

2.5 SRTP (Secure Real-Time Transport Protocol)

Real-Time Transport Protocol (RTP) [RFC 3550] is designed to carry data over an IP network, primarily over the UDP transport layer. RTP has a secure profile called Secure RTP [RFC 3711]. RTP is used for transport of real time data such as e.g. audio and video. SRTP can provide confidentiality, message authentication, and replay protection to that traffic. SRTP defines a format, specifies encryption algorithms to use and supplies a key derivation mechanism. The key derivation mechanism requires a master key eg. from MIKEY.

SRTP packet:



where (optional):

- MAC (Message Authentication Code) applies to RTP header and payload
- MKI (Master Key Identifier) tells the receiver which key to use. Compare with SPI in IPSEC SA.

SRTP protects the traffic on the application layer, and should as such be independent of the transport and network layer. These layers are not encrypted. All the protection mechanisms are implemented in the application which gives independence from the operating system.

2.6 MIME (Multipurpose Internet Mail Extensions)

Multipurpose Internet Mail Extensions (MIME) defined in [RFC2045] and [RFC2046] was originally designed to extend the capabilities of mail bodies by introducing a body structure. This enabled the transfer of other content than plain US-ASCII. MIME is now not only used to extend mail functionality but more as a general way of describing message bodies e.g. SIP. MIME defines a number of new headers among others:

Mime-Version: Content-Type: Content-Encoding: Content-ID: Content-Description:

In minisip the most used SIP/MIME headers are Content-Lengh: describing the length of the message body in bytes and Content-Type: describing how the message body should be interpreted. e.g:

Content-Type:application/sdp

In the above row the content type is application and the applications subtype is sdp, and the message body contains the Session Description Protocol. This is how SIP transports the SDP.

Content-Type: multipart/mixed; boundary=boun=_dry

The message body contains different body parts separated with the boundary boun=_dry. See example in section 5.2

2.7 S/MIME (Secure MIME)

S/MIME, as described in [RFC 3261], can be used to secure the content of the SDP inside the SIP message. Note that use of S/MIME requires the existence of some PKI solution or pre-shared secrets. Example of two types of secure MIME bodies for SIP:

- multipart/signed [RFC 2633] [netsec], is used for signing messages. The signature is held separately from the SDP message so even a recipient that do not support S/MIME can read the message.
- application/pkcs7-mime [RFC 2633] [netsec], is used both for encrypted messages and signed messages. The message is first signed then encrypted. This way the identity of the signer is kept a secret. The message is encrypted as an enveloped message i.e. encrypted with a random session key that is protected by the public key of the recipient.

The following is an example of an encrypted S/MIME SDP body within a SIP message from [RFC 3261]:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
        name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
        handling=required
**************************************************
* Content-Type: application/sdp                  *
*                                                *
* v=0                                            *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com  *
* s=-                                            *
* t=0 0                                          *
* c=IN IP4 pc33.atlanta.com                      *
* m=audio 3456 RTP/AVP 0 1 3 99                  *
* a=rtpmap:0 PCMU/8000                           *
**************************************************
```

Where the SIP header fields:

- Content-Type: -indicates the media type of the message body sent to the recipient.
- Content-Disposition: -describes how the message body is to be interpreted by the recipient

3 Existing solutions

Most of the work that has been done regarding SIP initiated secure media sessions is in conjunction with IP telephony (VoIP). Solutions where SIP initiates IPSEC for the media session have not been found. There are two main issues regarding SIP security.

3.1 Secure VoIP media session

There are some existing solutions regarding secure VoIP sessions but none of them uses IPSEC(ESP) as the security protocol.

Examples are:

- Cisco (www.cisco.com) has a solution for secure VoIP based on SRTP in CallManager 4.1.
- Skype (www.skype.com) has some proprietary solution for media security based on AES.
- Minisip (www.minisip.org) uses SRTP with MIKEY for key agreement.
- [sdescriptions] describes a way of establish security parameters for SRTP with a SDP attribute a=crypto. This attribute is not a key management protocol, a=crypto just conveys a set of parameters for SRTP. To my knowledge it do not exist any implementation of [sdescriptions].

3.2 Secure signaling

Securing the SIP signaling has nothing to do with the media session protection, but there is an interesting solution regarding the use of IPSEC described in [RFC 3329] and adopted by 3GPP [TS 33.203]. The main part of this is the definition of three new SIP header fields:

- security-client
- security-server
- security-verify

These fields are used to negotiate security between the user agent and the first hop proxy. The client tells the proxy its capabilities with security-client and the proxy tells its capabilities with security-server. The offers are confirmed with SIP:security-verify.

The security mechanisms that can be used are, as defined in RFC 3329, ("digest"/"tls"/"ipsec-ike"/"ipsec-man"/"token") to this 3GPP [TS 33.203] has made an extension for manually keyed IPSEC (ipsec-3gpp) that make it possible to exchange more IPSEC specific parameters e.g. SPI and a limited policy.

[RFC 3329] do not define how keying materials are exchanged for manually keyed IPSEC, it just assumes it is present at both peers.

This solution is of interest since it is a method of establish IPSEC connections as a part of a SIP setup, but is only used to secure SIP signaling.

4 Possible approaches to a solution

Encryption and authentication (ESP/AH) in the IPSEC connection will, in this thesis, use the existing Linux native IPSEC support as is, which is a port from KAME [KAME]. The solutions discussed below examine ways of exchanging IPSEC parameters to set up the connection. There are two scenarios:

1) Make it possible to protect only the traffic described in the SDP.
2) Make it possible to protect general traffic. This is the preferred scenario since it includes the first and does not limit the possibilities with IPSEC.

To this issue there is a number of restrictions that need to be regarded in this discussion. First there are some abstraction barriers that should not be crossed and some interesting characteristics of the involved technologies. Some of these restrictions are:

1. Where to transport the security parameters for IPSEC? (SIP vs. SDP)

   SIP is used for signaling and as SDP transporter of media session parameters. Information regarding SIP signaling and signaling related information should be carried in the SIP headers. Information that is related to the session that the user wants to establish between the involved user agents should be transported in SDP or some other description protocol transported by SIP.

   Since IPSEC is used by the media session and not the SIP signaling, the conclusion of this is that the IPSEC security parameters should be transported as a SIP payload and not as a SIP header.

2. Can the establishment of IPSEC SA at both user agents be regarded as the establishment of a multimedia session?

   Yes: The establishment of mutual SA is a preparation so that traffic can be exchanged even though no traffic might be exchanged, so an IPSEC session can be said to exist when there exist valid SA at both peers.

   No: Quote from the SDP specification [RFC 2327]: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session."

   The conclusion of this is that the IPSEC parameters should not be transported in the same session description as the description of the media, since IPSEC may protect more traffic than the set of multimedia senders and receivers defines. However, if IPSEC is supposed to protect *the exact traffic* described in the SDP, the IPSEC parameters may be transported in the SDP.

3. Only one session description is permitted in the SDP used by SIP.

   The SDP specification [RFC 2327] allows several session descriptions to be concatenated into a single SDP but the "offer/answer model" [RFC 3264] used by SIP only allows one session description per SDP. If more than one session description was allowed in the SDP one could have had the IPSEC parameters in a separate session description in the same SDP as the media description.

   The conclusion of this in combination with point 2 above is that the IPSEC parameters can not be transported in the same SDP as the media description, if we want to establish general IPSEC connections.

4. Do we need to have a media line (m=) in the SDP?

   If the SDP can be used without a media line the SDP can be used to transport IPSEC parameters independent from any media. E.g. if we want to establish a IPSEC connection but not a media session.

   No: The absence of the media line implies that the offerer wishes to communicate, but the streams for the session will be added at a later time. [RFC 3264]

   This might make it possible to use SDP for transport of IPSEC parameters but without any media description.

5. Although the number of session descriptions in a SDP used by SIP is limited to one, the number of payloads transported by a single SIP message is not limited to one [RFC 3261]. By the use of MIME multipart messages [RFC 2046], multiple payloads can be sent in one SIP message.

   This makes it possible to send the IPSEC parameters in a separate payload in SIP.

6. The possible use of a new secure audio/video profile to indicate that IPSEC should be used to secure a media stream (e.g., IPSEC-AVP, compare with Secure profile for RTP [RTP Profile]) is not possible since IPSEC as a protocol is independent from the media stream, and will generate a layer violation. See 4.2.2.

7. Assuming that SDP can not be used to exchange IPSEC parameters than another existing protocol must be used or new description protocol must be defined with the sole purpose of conveying security parameters.

4.1 SIP – IKE

IKE is maybe the most straight forward way of establishing IPSEC security parameters between the calling parties, since IKE [RFC 2409] is an existing protocol deployed and proven.

4.1.1 Running IKE after SIP call establishment

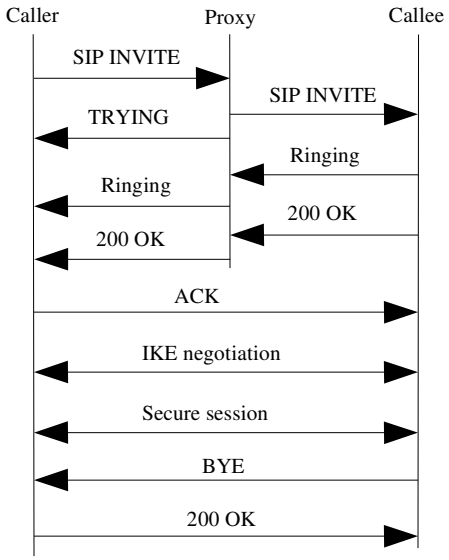


fig.5

For this scenario to work (at least) the following needs to be fulfilled:

1. Before the IKE session starts the IKE session initiator must know both parties IP addresses. Both addresses are known after the normal SIP setup signaling INVITE and 200 OK, or one could use SIP OPTIONS in a similar way as in [RFC 3329] to get hold of the IP addresses. See Appendix 6.
2. The caller needs a way to tell the callee that he wants to establish an IPSEC connection and the callee must tell the caller if he is able and willing, one could use a SDP attribute for this purpose. This could be done within the normal setup signaling SIP:INVITE and SIP:200 OK messages. This signaling must be protected from downgrade attacks, perhaps by the use of S/MIME.
3. The SIP UA needs a way to communicate with IKE daemon and IPSEC enabled OS kernel to initiate key exchange and manipulate IPSEC security policy, if the IPSEC security policy is not set manually by the user.
4. A Public Key Infrastructure (PKI) must exist or the caller and callee must have a pre shared secret or at least a secure way of exchanging a shared secret.

5. There must be a way for the caller and callee to make sure that the IPSEC connection is in place before the media session begins and a way to abort the call if the IPSEC connection is not present or faulty.

Advantage of using IKE for key exchange

1. Use of already existing and proven mechanism (IKE) with all its functionality for the key exchange.
2. Low impact on SIP UA implementation since the key exchange protocol does not need to be implemented by the user agent.

Disadvantage

1. Increased number of round trips since it does not use any of the SIP signaling packets for the key exchange. IKE adds 6 or 9 packets depending on IKE mode, see section 2.3.4.
2. The use of SIP:OPTIONS would add even more round trips since the SIP:INVITE, SIP:200 OK cycle or some other negotiation method are needed for the media stream.
3. Increased delay in the setup of a call, since IKE does not start the negotiation until one sends a packet that matches the security policy.

To ensure that the IPSEC connection is established before the media session starts one can use provisional responses(SIP:1xx) before the 200 OK message. This can also be used for telling the caller that the callee do not support IPSEC or for exchanging any other relevant information. Provisional responses have not been further evaluated in this thesis.

4.1.2 Carrying IKE messages in SIP

SIP signaling uses three packets to establish a session INVITE, 200 OK and ACK. IKE aggressive mode also uses three packets and one could think of piggybacking IKE on SIP. This is however not possible since IKE requires the knowledge of the callee's IP address before the signaling begins, an information that we can not assume that the caller has before the SIP signaling is complete, unless i.e. SIP OPTIONS has been used to exchange IP-addresses.

4.1.3 IKE independent from SIP

IKE can of course be used independent from SIP, as it is used in most cases today. But this requires the knowledge of the IP addresses of the calling parties before the call is made. This case has been studied in [ImpactofKey].

4.1.4 Running IKE as a call establishment pre-condition.

IKE could be a call establishment pre-condition in line with RFC 3312 [RFC 3312]. It is an interesting idea, however this came up late and has not been studied further.

4.2 Key exchange in SDP attribute of SIP signaling

Another solution would be to use a SDP attribute to distribute necessary IPSEC parameters. It may be described as follows:

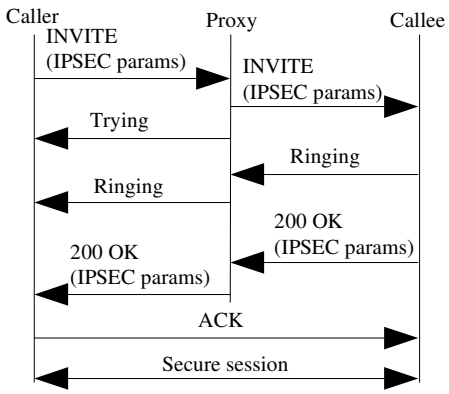


fig.6

There are currently three SDP attributes used for distributing keying materials: The SDP "k", SDP "a=crypto" and SDP "a=key-mgmt" attributes. The main benefit of using SIP signaling to exchange IPSEC parameters is that no extra round trips are needed for the key exchange.

4.2.1 SDP k

SDP k=<method>:<encryption key>

The k attribute is described in [RFC 2327].
This attribute is very limited in its scope since it is only used to convey an encryption key for RTP, and has no means to convey other cryptographic parameters. This characteristic makes this attribute not suitable to convey IPSEC parameters since they consists of more than just a key and the k attribute is only defined as a general means for securing RTP. The "k" attribute is in clear text and relies on other means for protection, e.g., S/MIME.

4.2.2 SDP a=crypto

SDP a=crypto:<tag> <crypto-suite> <key-params> [<session-params>]

The a=crypto attribute is described in [sdescriptions].
This attribute is more general than the k attribute described above. It specifies a way to signal and negotiate cryptographic parameters for media streams in general. It is possible to define crypto-attribute parameters to convey IPSEC parameters but this is not a very good idea because:

1. a=crypto only has a meaning when a secure transport protocol is indicated (e.g., "RTP/SAVP" or "RTP/SAVPF" as described in [RTP Profile]) in the SDP media (m=) line. That is if there is a media line indicating a secure profile for RTP than the a=crypto attribute is used. If there is a media line that indicates a non secure profile for RTP the a=crypto is not needed and not used even if present. See also the general requirements in the beginning of this chapter.
2. If there is no media line (m=) at all this attribute has no meaning. Since there has to be at media line that indicates a secure profile for RTP if the a=crypto should have a meaning. If the attribute had a meaning without the media line, it could have been used to convey IPSEC parameters for general IPSEC connections.

To make a=crypto work for IPSEC there has to be a change in the scope of the a=crypto attribute. The attribute has to independent of the media line and always be used, if present, regardless if the media line indicates a secure profile or not. This because IPSEC can be used even if the media line indicates a non secure profile for RTP.

4.2.3 SDP a=key-mgmt

SDP a=key-mgmt:<prtcl-id> <keymgmt-data>

The a=key-mgmt attribute is described in [kmgmt]

This attribute specifies a way of exchanging messages generated by a key management protocol. Since it is up to the key management protocol to transport the cryptographic parameters this attribute can be used to exchange parameters for IPSEC. But as with a=crypto, described above, the a=key-mgmt is very related to the media transport because of the same reasons as in 4.2.2.

The attribute key-mgmt combined with the key management protocol MIKEY is used in the current version of minisip to secure the media session with SRTP.

4.2.4 SDP a=”new”

The attribute a=key-mgmt is a good approach to exchange parameters for IPSEC since it allows the use of a key exchange protocol e.g. MIKEY, which can be adopted to exchange parameters for IPSEC, but the problem is the connection between the a=key-mgmt attribute and the media transport. To solve this issue a new attribute could be introduced, here called "new".

Reasons to the introduction of a new attribute for cryptographic parameter exchange:

1. The reason for introducing a new attribute is independence from the media transport since IPSEC is not a transport protocol. The existing solutions focus on securing media streams, to use IPSEC one must "forget" about the application generating the traffic and only think of traffic as source address, destination address, UDP/TCP and ports.
2. This new attribute should be able to work independently or in conjunction with the media and their transport mechanisms.
3. The basic functionality would be like a=key-mgmt with the exception of the relationship to the media.
4. If just a particular type of traffic (UDP/TCP, ports) and/or specific source and destination addresses should be protected this is handled by the IPSEC policy. Specifying proposed policy can for instance be done by placing the attribute on media or session level in the SDP.

This is the best of the "SDP" alternatives since it would be able to negotiate all parameters in one round trip and still be independent of the possible media.

The SDP attribute can be described as follows:

SDP a=”new” <prtcl-id> <keymgmt-data>

In consideration of the constraints in the beginning of section 4 my suggestion is that the following functionality apply:

1. a="new" is a session level parameter indicating that IPSEC should be used
2. if IPSEC should be used to secure more traffic than described in the media session description a="new" must not be in the same SDP as the media description. If IPSEC should be used to secure traffic in general the a="new" must be in a SDP that has no media line (m=).
3. if the a="new" attribute is in the same SDP as the media session description the IPSEC policy must be defined to match the traffic described in that media session description exactly.
4. a="new" parameters is extended compared to a=key-mgmt so that the syntax will be a="new" <prtcl-id> [<keymgmt-data>], making the <keymgmt-data> optional for some <prtcl-id>, so that it can used for protocols that do not convey <keymgmt-data> in the attribute e.g IKE.
5. if no IPSEC SA is to be created a="new" must not be present.

However the SDP is so closely related to transport of media specific information so introducing a new attribute that do not have any relation to the media is in my current opinion to put things in the SDP that do not belong there. Unless one want to have a close relation to the specific media i.e. specifying a policy that matches the media description, the attribute should be should be put elsewhere than in the SDP.

4.3 SIP-MIME-MIKEY

In this solution all IPSEC parameters are transported in a MIKEY message. The MIKEY message is carried as a MIME payload in SIP. This solution is described in detail in section 5. This is also the solution that has been implemented into *minisip*.

5 Implemented solution SIP-MIME-MIKEY

In this chapter the chosen solution is described. The implementation description of this solution is in Appendix 2.

This solution and the implementation is a proof of concept and is as such not to be regarded as anything else than just a proof of concept.

The chosen solution can be described as follows:

- With the use of a MIKEY message that is transported in a MIME multipart body part, two IPSEC SA and two IPSEC policy entries are set in each client host. One SA and policy for the outgoing traffic and one for the incoming in each end . This enables IPSEC transport mode for all traffic between these two hosts. All this is done by the initiator with no more interaction than the setting of "use_ipsec" in the configure file of *minisip*. It should however be possible to select the protected traffic based on the information in the SDP, see Future work chapter 8.

The decision why to go for the chosen solution is summarized in the following statements:

- Usage of MIKEY

  IPSEC parameters should be exchanged in one round trip in a similar way as in the case of SRTP/MIKEY. The choice of one round trip makes MIKEY a good solution since this was one of the design goals of MIKEY, and has proven a good solution for SRTP in *minisip*. Hence MIKEY will be used and adopted to carry IPSEC parameters in a similar way as it can carry SRTP parameters. The IPSEC profile for MIKEY is described in section 5.1.

- Carry MIKEY outside of the SDP

  In the SRTP case the MIKEY message is carried within the SDP. This is however not a good option in the IPSEC case because IPSEC protects the traffic on the network layer, and what traffic to protect is based on IP, transport protocol and ports, and has nothing to do with the media session. IPSEC is independent from the media stream and one should be able to use both IPSEC protection as well as SRTP since they are independent and add protection to the traffic at different layers. The SDP is used to describe media sessions. If one wants to protect just the traffic described in the SDP, that information can be extracted from the SDP when IPSEC policies are generated. If the the MIKEY message is placed in the SDP it will be bound to that specific media session and the traffic it generate, and the flexibility of IPSEC will be lost.

- Carry MIKEY in MIME

  The SIP RFC [RFC3261] gives an opening to the problem that the IPSEC MIKEY can not be carried in the SDP by allowing multipart MIME [RFC 2045] [RFC 2046] as SIP payload. The SDP is carried in SIP as a MIME message with content type application/sdp. This scenario will carry the SDP, with the media description, as a multipart MIME body part with content-type application/sdp in a MIME multipart message with content-type multipart/mixed. Within the multipart message another body part will contain a content-type application/mikey containing the MIKEY message with necessary parameters for the IPSEC setup. This way the two body parts application/sdp and application/mikey will meet the requirement of being independent. This is described in section 5.2.

For this to work there has to be an addition to current standards. First MIKEY must be adopted to carry IPSEC parameters (see section 5.1) and the MIME content-type application/mikey defined (see section 5.2). The used IPSEC implementation is commented in section 5.4. There should also be some additions to the SIP logic of the user agent. The SIP logic is the "rule base" SIP uses to know the reactions to specific events. E.g, how should the SIP UA react when receiving a SIP INVITE with a payload of Content-type: application/mikey, or if the UA do not understand the request. See section 5.3.

5.1 IPSEC profile for MIKEY

MIKEY is described in [RFC 3830] for use with SRTP and a Crypto Session(CS) ID map type, SRTP-ID and Security policy with parameters is defined for SRTP. To use MIKEY with IPSEC (ESP/AH) a new CS ID map type must be defined, with the corresponding CS ID map info and a security policy with parameters and security policy protocol type, as stated in RFC 3830 section 4.2.9 [RFC 3830]

IPSEC is dependent on the IP version. For this implementation I have suggested values for IP version 4 (IPSEC4). All values and formats for IPSEC4 is to be defined. All values and formats defined here are for testing purposes in this implementation and proper standardization and assignment from IANA is required for future use. The format of the MIKEY CS ID attributes and MIKEY security policy for IPSEC below is written here in the same way as they are described for SRTP in [RFC3830].

The CS ID map contain Crypto Sessions (CS), where each CS describes a SA. SPI, source address, destination address of the SA are transported here. The rest of the parameters for the SA are transported in the MIKEY security policy payload. Each CS requires one MIKEY security policy payload, but it can be the same for all CS.

5.1.1 CS ID map

The CS ID contains information on how to protect IP traffic with a specific source address and destination address. These CS ID's are kept in the CS ID map (see fig.8) and the field "CS ID map type"(see fig.7) defines how the "CS ID map info" should be interpreted.

CS ID map type	Value
SRTP-ID	0
IPSEC4-ID	7

fig.7 CS ID map type (SRTP-ID is standardized in [RFC 3830])

CS ID map info IPSEC4-ID

Policy_no_1 (8bits)	SPI_1 (32 bits)	spiSrcaddr_1 (32 bits)	spiDstaddr_1 (32 bits)
Policy_no_2 (8bits)	SPI_2 (32 bits)	spiSrcaddr_2 (32 bits)	spiDstaddr_2 (32 bits)
Policy_no_#CS (8bits)	SPI_#CS (32 bits)	spiSrcaddr_#CS (32 bits)	spiDstaddr_#CS (32 bits)

fig.8 CS ID map info IPSEC4-ID

5.1.2 Security policy payload for IPSEC4

The MIKEY security policy payload contains security parameters for the security protocol. Values of the different policy types(see fig.10) are values defined in [RFC 2367]. Parameters defined here is not the complete set available for IPSEC. These are the parameters that are essential for the implementation in this thesis. The Protocol type field (see fig.9) value defines how the policy types should be interpreted.

Protocol type	Value
SRTP	0
IPSEC4	7

fig.9 Protocol type values (SRTP is standardized in [RFC 3830])

Policy Type	Meaning	Possible values
0	SATYPE	SATYPE_ESP = 3
1	MODE	TRANSPORT = 1
2	FLAGS	sequential padding for ESP = 0
3	EALG	3DESCBC = 3
4	EKEYL	depends on cipher used
5	AALG	MD5HMAC = 2; SHA1HMAC = 3
6	AKEYL	depends on MAC used

fig.10 Policy type

5.2 Content-Type application/mikey

Multipurpose Internet Mail Extensions (MIME) is defined in [RFC 2045] [RFC 2046] and SIP messages carry MIME payloads. As discussed in section 2.6 the SDP with the media description is carried in a MIME message with Content-Type application/sdp and the MIKEY message for IPSEC is carried in a Content-Type application/mikey. The information in the body part application/mikey is a base64 encoded MIKEY message as described in [RFC 3830].

Example of INVITE message dump generated by minisip with multipart payload.

```
INVITE sip:orrblad@ssvl.kth.se;user=phone SIP/2.0
From: <sip:joachim@ssvl.kth.se;user=phone>;tag=952824928
To: <sip:orrblad@ssvl.kth.se;user=phone>
Call-ID: 196120334@192.16.125.178
CSeq: 401 INVITE
Contact: <sip:joachim@192.16.125.178:5050;user=phone;transport=UDP>;expires=1000
User-Agent: Minisip
Content-Type: multipart/mixed; boundary=boun=_dry
Via: SIP/2.0/UDP 192.16.125.178:5050;branch=z9hG4bK697135402
Content-Length: 675

--boun=_dry
Content-type: application/mikey

AQAFgD+LoOoCBwAhXlqzAAAAALJ9EMAAAAAAALJ9EMAAAAAACgDFxW0OT9+PRwsABwAVAAEDAQEBAgEAAw
EDBAEYBQEDBgEQARDy6e7T10Z+aPj5NTNI9hFzAAAAxHbDvNnT1S9tBbcF4e9kx8lexfu0lCU7Zgf8t4wmQq+1rDq8cYcAz
O2YMdmBfNSfU9XTyOUxMPEudWwLBT22WW6rBNfmWnOil0KGE06pbDEKme84jgSHwV8Aj6nKUJ+3ZW9b4Mx1+qZgErl
vU1CdF6OVe1wbziQszV3QfvzqDs3l6OCmTZtirRuJfY+m8vobF38q4XWaDINnJhEBZi00hVMvpVL35aR7YZMRvzanuvrW1gxB
oVVqICoS5otKgrdqLG3FwpsA+OGp7kqm2tWQnL3WALOSH8PsNz8=

--boun=_dry
Content-type: application/sdp

v=0
o=- 3344 3344 IN IP4 192.16.125.178
s=Minisip Session
c=IN IP4 192.16.125.178
t=0 0
m=audio 33730 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1


--boun=_dry--
```

5.3 SIP Logic

How to handle a MIME multipart as payload in SIP is not well defined in [RFC3261]. The way to treat a multipart message depends on the subtype (mixed, digest, parallel, alternative). In this implementation the multipart/mixed is used meaning that the different body parts should be handled independent. But if one of the body parts is not accepted or contains errors the call will be rejected. The SIP logic will be discussed more under section 7 Future work.

5.4 IPSEC

The implementation in this thesis make use of the native Linux IPSEC support available in kernel versions ≥2.5.47 and and 2.6.* and has been tested in this thesis on Linux kernel 2.6.7, 2.6.8 and 2.6.10. An implementation of PF_KEY [RFC 2367] called libipsec is used to manage the IPSEC kernel. Libipsec is part of the ipsec-tools [tools] released for Linux. Ipsec-tools is a port from KAME's [KAME] IPSEC utilities.

6 Measurements

This measurement is a follow up on a previously done measurement at KTH, "Call establishment delay for secure VoIP". The study was done by Johan Bilien, Erik Eliasson and Jon-Olov Vatn at IMIT/TSLab KTH [callest]. The goal is to measure the establishment delays of a VoIP call in three cases. The first two (no security and MIKEY/SRTP) were done in the previously measurement and are redone here as reference. The third is related to the outcome of my master thesis, the combination of MIKEY and ESP.
The previous measurement came to the conclusion that the establishment delay for a secure VoIP call is insignificant for a human user. One could expect that the result would be similar with MIKEY/ESP, and in most aspects they are, but as shown below the system calls to set the IPSEC SA in the native Linux IPSEC implementation is very time consuming. All SIP signaling in this test is using UDP as transport protocol and pre-shared secret as MIKEY authentication.

The testbed(fig.11) was setup to resemble the testbed used in the previous case. All hosts were installed with Debian distribution of Linux 2.6.10. Debian GNU/Linux 3.1.

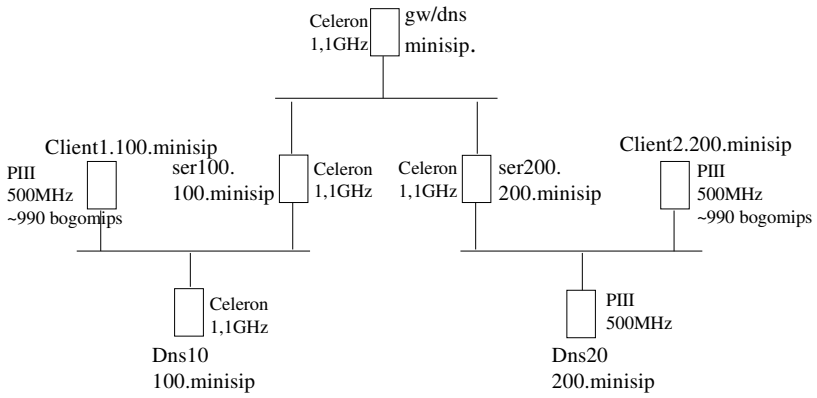


fig.11 The testbed

The *calling delay* is the time from when the caller has dialed the callee until the caller receives SIP Ringing message, below referred to as ∂2. The *answering delay* is the time from when the callee picks up the phone until the callee receives the SIP ACK message, below referred to as ∂7.

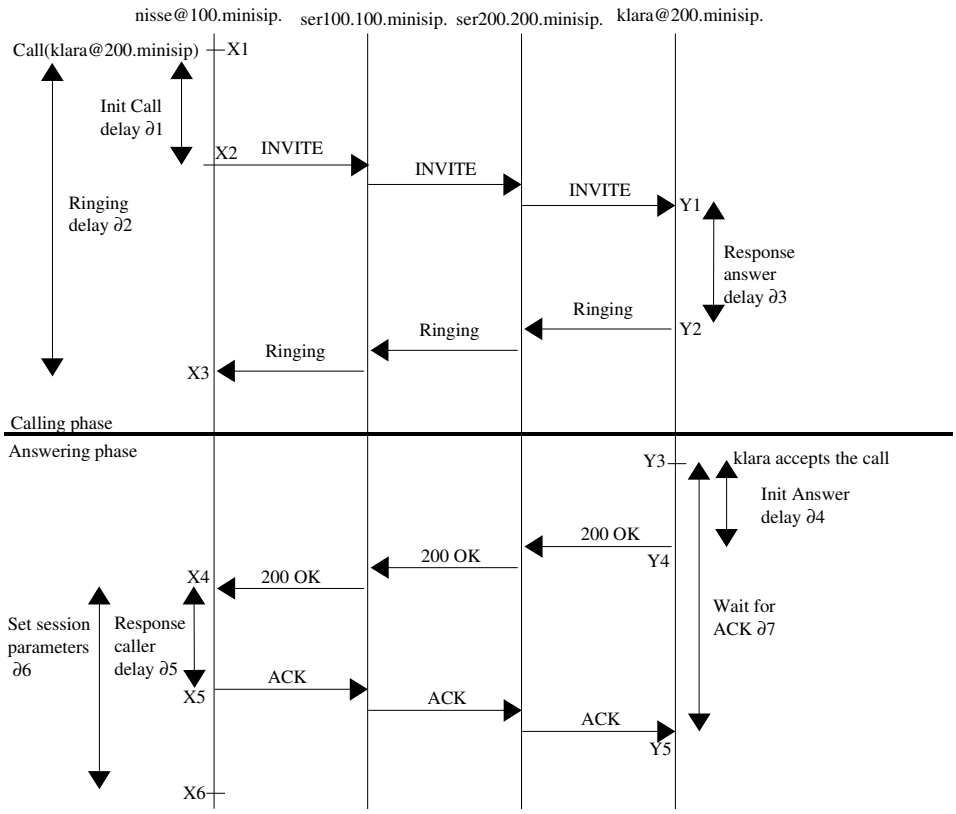


fig.12 SIP trapezoid with timestamps and calculated intervals.

To do the measurement eleven timestamps were added in source code of minisip. X1-X6 and Y1-Y5, see fig.12 above. ∂1-∂7 are then calculated as the time difference between the corresponding timestamps. Since the ∂-values are relative values between two timestamps on the same host the accuracy should be sufficient, and the values of *No security* and *MIKEY/SRTP* in this measurement conforms to [callest]. The raw data for the measurement is presented in Appendix 5.

In the figure above X1-X6, Y1-Y5 and ∂1-∂7 are described, but are also further explained below:

- ∂1 - The time from the point when the caller makes the call to the point where the INVITE message leaves the user agent.
- ∂2 - The time from the point when the caller makes the call to the point where his/her phone rings. This is the calling delay.

- ∂3 - The time it takes for the callee's user agent to process the INVITE and produce Ringing.
- ∂4 - The time from the point when the callee accepts the call to the point where the 200 OK message leaves the user agent.
- ∂5 - The time it takes for the caller's user agent to produce ACK after receiving the 200 OK.
- ∂6 - The time it takes for the caller's user agent to process the 200 OK and set the session parameters.
- ∂7 - The time the callee's user agent waits until receiving the ACK. This is the answering delay.

	Calling phase			Answering phase			
	∂1[mS]	∂2[mS]	∂3[mS]	∂4[mS]	∂5[mS]	∂6[mS]	∂7[mS]
No security	4.6	23.2	8.0	2.4	3.0	4.1	12.4
MIKEY/SRTP	7.5	29.0	9.9	3.1	3.1	4.8	13.5
MIKEY/ESP	7.8	32.3	9.8	691.3	3.2	704.0	704.4

fig.13 Result of the measurement. Average delays. (8 samples)

Note that ∂4 and ∂6 (∂7) in MIKEY/ESP includes the setting of SA to the kernel, 2 calls to the function pfkey_send_add from libipsec in both ∂4 and ∂6. Each function call takes about 330mS. This value will decrease on a faster computer but is still measured in tenth of milliseconds.

In the current implementation of Minisip ∂7-∂4 needs to be greater than ∂6 otherwise the ACK reaches the callee before the caller has set the session parameters. This will result in the loss packets in the beginning of the call, as with the case with ESP in the figure below. The values below is however dependent on the IPSEC implementation, and since I only used Linux native I can not say that this is the case for all IPSEC implementations.

	∂7-∂4[mS]	∂6[mS]
No security	10.0	4.1
MIKEY/SRTP	10.4	4.8
MIKEY/ESP	13.1	704.0

fig.14 Time to set parameters

This also implies that the actual answering delay is not ∂7 for MIKEY/ESP but ∂4 + ∂6 + the time for the 200 OK message to traverse the network, which is > ∂4 + ∂6.

7 Conclusions

SIP initiated IPSEC works, but there are some reservations. My addition to the Minisip user agent implementation makes it possible to establish IPSEC connection between two hosts under the condition that the two hosts have a shared secret or are able to verify each others certificates. This is the same as with SRTP and MIKEY.

Section 7.1 compares ESP with SRTP regarding the protection of VoIP. MIKEY and Minisip are covered in section 7.2 and 7.3 and section 7.4 is about the implications on SIP.

7.1 ESP vs SRTP

Although my recommendation is that SRTP is used to protect VoIP calls, the SIP initiated IPSEC can of course be used to protect traffic in general. The reason I prefer SRTP for VoIP is based on the following:

- To use IPSEC one has to communicate with some IPSEC software that usually is located in the kernel. This arises a number of issues. First of all you normally has to be superuser to communicate with the kernel, and the communication takes place between the user space and the kernel space. This issue should be possible to solve since this is the case for almost anything an application does (I/O, system calls) however it is more complex than for SRTP. The user agent also becomes totally dependent on which IPSEC implementation and/or operating system it is running on. The current implementation has been made for the native Linux IPSEC support and has only been tested with Linux kernel versions 2.6.7, 2.6.8 and 2.6.10. SRTP does not have these dependencies.

- As the observant may have noticed already that another problem became obvious in section 6 Measurements. In the current implementation of Minisip ∂7-∂4 needs to be greater than ∂6 otherwise the ACK reaches the callee before the caller has set the session parameters. This will result in the loss of packets in the beginning of the call, as with the case with ESP in the figure 14. When the callee answers the call there will be no one in the other end for the first 0.7 seconds. The values in figure 14 is however dependent on the IPSEC implementation, and since I only used Linux native I can not say that this is the case for all IPSEC implementations.

  I do not know the reason why this function call takes this long. I asked the IPSec tool development team and got the answer that it is "because of some system call", but I believe it does not need to be this long. One way of reducing the impact of this delay would be to put the MIKEY response in a provisional

response(PRACK) as mentioned in [callest]. This would increase the ringing delay, but decrease the more time critical "answering delay".

- In order to make Minisip or any other user agent usable in a wider perspective the user agent must have the ability to traverse NAT (Network Address Translation) and firewalls (FW). Adding ESP complicates these two issues because NATs and firewalls usually make decisions on IP-addresses, transport protocols and ports and with ESP we hide the transport protocol. To simplify this issue Linux native IPSEC provides FW/NAT traversal support by the use of an extra UDP headers between the IP header and the ESP header. This will however add extra overhead and makes ESP look much like SRTP but with larger overhead.

  RTP: IP | UDP | RTP header | RTP payload |

  SRTP: IP | UDP | RTP header | RTP encrypted payload | MKI | MAC |

  <------- encrypted ------->

  ESP: IP | UDP | ESP | UDP | RTP header | RTP payload | ESP tail |

  <---------- encrypted ---------->

  The RTP header + RTP payload is the same size as RTP header + RTP encrypted payload

  This is however only a problem if there exists NAT/FW. The introduction of IPv6 may remove the NATs but the FW will probably still be there. The decreased payload, because of the added UDP header, is probably not an issue since VoIP packets are relatively small and normally do not use the available payload space.

- IPSEC protects the traffic host to host i.e. IP-address to IP-address and do not give application to application security. See section 2.3.1.

IPSEC may still by valuable since it is well recognized and has the ability to protect general traffic, not just the media streams. This feature makes it possible to establish VPN connections between two minisip UA so that all traffic exchanged between these UA's are protected.

7.2 MIKEY

MIKEY is designed in a very flexible way making it possible to convey security parameters for almost any security protocol. The adoption of MIKEY for IPSEC was rather straight forward. All changes done to MIKEY are described in section 5.1.

7.3 Implementation of *minisip*

The measurements in section 6 shows that setting session parameters can be time consuming. Possible timing problems can arise if the user agents are running on slow computers and the network delay is low, even with SRTP and no security. This can be avoided if the caller's user agent sends the ACK after setting the session parameters in the same manner as the callee's user agent sending 200 OK after setting the session parameters. The cost of doing this is the time that the ACK needs to traverse the network from the caller to the callee and the benefit is that when the ACK reaches the callee both parties are ready for the conversation. Another problem also arises, in the current implementation of *minisip*. If the ACK is sent directly and not via the proxy's it will be discarded by the callee since the callee expects IPSEC traffic and the caller has not established his parameters when he sends the ACK. This will result in a new 200 OK from the callee since no ACK got through, and the caller will send a new ACK. This cycle will continue until the ACK reaches the callee.

7.4 SIP

In this thesis I show that it is possible to establish IPSEC with the help of SIP and MIKEY. To this there are some interesting questions about what happens next in the SIP call.

- When the call is terminated the IPSEC policy and SA should be removed. The party that sends the BYE can remove the items when he receives the final 200 OK but the other party can not remove them until after the timeout after the final 200 OK has been sent, because he does not know if the other party has received the 200 OK. If the signaling is relayed over the SIP proxy's this is not an issue, then the policy and SA can be removed without and concern of timeouts. But of this we can not be sure since the final BYE and 200 OK can go directly between the caller and the callee and in that case IPSEC must be configured at both ends until all signaling is done and all timeouts are finished. Today they are removed after the final timeout (see fig.15 below) both at the caller and the callee, which mean that the current implementation do not have this problem.

- My current implementation of the MSipIpsecAPI in minisip uses one MSipIpsecAPI object for each call which can be somewhat of a problem in some cases. There is no coordination between different calls regarding IPSEC requests to the kernel which will create unwanted effects when:
    - Two calls want to set identical policies to the kernel.
    - Two calls to the same destination will result in two different SA and if they are the same (with the exception of SPI and keys) there will be a conflict since the kernel will not know which SA to choose for the outgoing traffic.

  It would be better to have one MSipIpsecAPI object for all calls from the same client, but the problem will arise again when two clients are running on the same host. This because there is only one kernel.

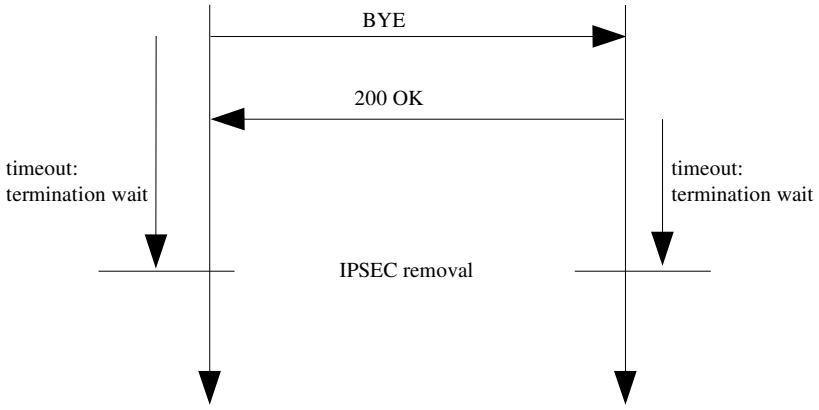


fig.15

8 Future work

- SIP Logic

  If this study will have a life after this report there should be more definitions and adaption of the SIP logic to handle IPSEC/MIKEY messages in a more general way. This implementation takes a limited number of scenarios into account. Some additional scenarios might be:

  - If the responder does not support multipart MIME.

  - The responder supports IPSEC but does not want to use it.

  - How should the UA respond to different subtypes of MIME content-type multipart/*?

  - Where and when and which error messages should be sent in case of errors, maybe just rejecting the call is not good enough.

- Multiple calls using IPSEC

  Multiple calls should work as long as they are not to the same destination(same policy and SA), but if they are how should one proceed to reuse already established SA and/or policy, and make sure that they are not removed when one of the calls is terminated.

  When one call is terminated and a new call is made to the same destination before the IPSEC policy and SA are removed, during the termination timeout, we get a scenario similar to that above. One possible solution would be to move the MSipIpsecAPI from the session level of SIP to a more global level in *minisip* so that the one MSipIpsecAPI object can keep track of all IPSEC parameters for all calls.

  What if there are multiple *minisip* clients on the same host? How should all of them be able to use IPSEC and avoid conflicts? There is only one SA-database and one IPSEC security policy database per kernel which should contain unique entries only.

- IPSEC security policy

  IPSEC security policy in the current implementation can not be changed, it will always be: -Protect all traffic between the two IP-addresses mentioned in the SA. How should the policy be handled? In IKE the IPSEC security is set manually, this can not be done in this case since no IP-addresses are known before the call setup. It should be possible to decide the IPSEC security policy based on information from the SDP. For this to work the information from the SDP must be given to the MSipIpsecAPI object that

will construct the policies. Entries in the MIKEY *Security Payload* could be used as a way of exchange IPSEC security policies, another is the use of MIKEY *General Extension Payload*.

- NAT/FW support

  Minisip should be able to use the IPSEC NAT support. All functions needed to apply this functionality exists in the libipsec library and should be called from the MSipIpsecAPI. An addition to the MIKEY policy types (see section 5.1.2) must be defined stating that IPSEC NAT support should be used.

- Mobility support has not been addressed in this thesis. Since users may like to roam in the middle of a conversation, addressing support to handle change of IP address during a call is an interesting topic left as future work. Handling mobility when security is accomplished via IPSEC as compared to SRTP may be a bit more difficult since the IP address is used to identify the IPSEC SA and security policy. Ideas to build future work upon may involve using Mobile IP (MIPv4[RFC3344], MIPv6[RFC3775]), or SIP application layer mobility[sipmob] in the following ways[1]. With Mobile IP, no additional mobility support would be required in *minisip*, but this would lead to some additional overhead for the data packets. The data packets would either need to be tunneled via the home agent, or in the case of MIPv6 with route optimization, the home address(es) of the mobile node(s) needs to be carried in header extensions. In contrast, if SIP mobility is used there will be no packet expansion, but the mobile user agent (*minisip*) needs to send a SIP re-INVITE[sipmob], and containing a new MIKEY INIT message with the updated IPSec parameters.

- MIKEY

  More IPSEC specific parameters should be conveyed by MIKEY. There are more functionality to IPSEC than is used in this thesis. To be able to use these functionalities there must be a way to exchange their parameters. To do this an addition to the MIKEY policy types (see section 5.1.2) must be defined.

  There is no re keying done in the current implementation of *minisip.* Rekeying with MIKEY must exist if longer sessions are to be maintained. The implementation of this is left for future work.

  To make MIKEY work for IPv6 IPSEC at least MIKEY *CS ID map info* and *CS ID*

1 Private discussion with Jon-Olov Vatn, March 2005

map type for IPv6 IPSEC has to be defined.

- Miscellaneous

  Solve the issue regarding the fact that you have to be superuser to set IPSEC parameters. Normally you do not want to run applications as root. This should not be to hard to solve as previously mentioned in 7.1.

  How does the use of different encryption algorithms influence the ability to recover packet loss.

  User preferences. In this implementation the caller sets a number of default values, but this should be something that the user should be able to choose. The user needs to edit the minisip configuration file (.minisip.conf) to enable IPSEC for an outgoing call. This should of course be possible to do from the GUI.

  It should be possible to use other IPSEC implementations than Linux native, e.g., Freeswan. A port to Ms Windows would also be desirable.

9 References

[callest] Johan Bilien, Erik Eliasson and Jon-Olov Vatn. Call establishment delay for secure VoIP. WiOpt'04, Cambridge UK, March 2004

[ImpactofKey] Mohan Krishna Ranganathan and Liam Kilmartin. Investigations into the Impact of Key Exchange Mechanisms for Security Protocols in VoIP Networks. In Proceedings of the 1st IEI/IEE Telecommunication Systems Postgraduate Research Symposium, Dublin, Ireland, 2001

[ikev2] Charlie Kaufman. Internet Key Exchange (IKEv2) Protocol. Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-17.txt. Work in progress. September 2004.

[KAME] http://www.kame.net "KAME Project is a joint effort of six companies in Japan to provide a free IPv6 and IPsec (for both IPv4 and IPv6) stack for BSD variants to the world."

[kink] M. Thomas, J. Vilhuber. Kerberized Internet Negotiation of Keys (KINK). Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-kink-kink-06.txt. Work in progress. December 2003.

[kmgmt] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, Ericsson. Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP). Internet Engineering Task Force,http://www.ietf.org/internet-drafts/draft-ietf-mmusic-kmgmt-ext-11.txt. Work in progress. April 2004.

[netsec] Charlie Kaufman, Radia Perlman, Mike Speciner. Network Security PRIVATE Communication in a PUBLIC World second edition. ISBN 0-13-046019-2. 2002

[PK01] Perlman, R., and Kaufman, C., "Analysis of the IPsec key exchange Standard", WET-ICE Security Conference, MIT,2001, http://sec.femto.org/wetice-2001/papers/radia-paper.pdf.

[RTP Profile] Joerg Ott, Elisabetta Carrara. Extended Secure RTP Profile for RTCP-based Feedback (RTP/SAVPF). Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-avt-profile-savpf-01.txt. Work in progress. July 2004.

[RFC 2045] N. Freed, N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2045.txt. Nov 1996

[RFC 2046] N. Freed, N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, RFC 2046. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2046.txt. Nov 1996

[RFC 2327] M. Handley, V. Jacobson. SDP: Session Description Protocol, RFC 2327. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2327.txt. April 1998

[RFC 2367] D. McDonald, C. Metz, B. Phan. PF_KEY Key Management API, Version 2, RFC 2367. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2367.txt. July 1998.

[RFC 2401] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol, RFC 2401. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2401.txt. Nov 1998.

[RFC 2402] S. Kent, R. Atkinson. IP Authentication Header, RFC 2402. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2402.txt. Nov 1998.

[RFC 2406] S. Kent, R. Atkinson. IP Encapsulating Security Payload (ESP), RFC 2406. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2406.txt. Nov 1998.

[RFC 2409] D. Harkins, D. Carrel. The Internet Key Exchange (IKE), RFC 2409. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2409.txt. Nov 1998.

[RFC 2633] B. Ramsdell. S/MIME Version 3 Message Specification, RFC 2633. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2633.txt. June 1999.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. SIP: Session Initiation Protocol, RFC 3261. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3261.txt. June 2002.

[RFC 3263] J. Rosenberg, H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3263.txt. June 2002

[RFC 3264] J. Rosenberg, H. Schulzrinne. An Offer/Answer Model with the Session Description Protocol (SDP), RFC 3264. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3264.txt. June 2002

[RFC 3312] G. Camarillo, W. Marshall, J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP), RFC 3312. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3312.txt.October 2002

[RFC 3329] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, T. Haukka. Security Mechanism Agreement for the Session Initiation Protocol (SIP), RFC 3329. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3329.txt. January 2003

[RFC 3344] C. Perkins, Ed. IP Mobility Support for IPv4, RFC 3344. Internet Engineering Task Force, ftp://ftp.rfc-editor.org/in-notes/rfc3344.txt. August 2002

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, RFC 3550. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3550.txt. July 2003.

[RFC 3711] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. The Secure Real-time Transport Protocol (SRTP), RFC 3711. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3711.txt. March 2004.

[RFC 3775] D. Johnson, C. Perkins, J. Arkko. Mobility Support in IPv6, RFC 3775. Internet Engineering Task Force, ftp://ftp.rfc-editor.org/in-notes/rfc3775.txt. June 2004.

[RFC 3830] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman. MIKEY: Multimedia Internet KEYing, RFC 3830. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2830.txt. Aug 2004

[RFC 3986] T. Berners-Lee, R. Fielding, Day Software, L. Masinter. Uniform Resource Identifier (URI): Generic Syntax, RFC 3986. Internet Engineering Task Force, ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt. January 2005

[sdescriptions] Flemming Andreasen, Mark Baugher, Dan Wing, Cisco Systems. Session

Description Protocol Security Descriptions for Media Streams. Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdescriptions-07.txt. Work in progress. July, 2004

[sipmob] Henning Schulzrinne and Elin Wedlund. Application Layer Mobility Using SIP. ACM SIGMOBILE Mobile Computing and Communications Review, number 3 2000.

[tools] http://ipsec-tools.sourceforge.net "IPsec-Tools is a port of KAME's IPsec utilities to the Linux-2.6 IPsec implementation."

[TS 33.203] 3rd Generation Partnership Project. Technical Specification Group Services and System Aspects 3G security. Access security for IP-based services (Release 6). http://www.3gpp.org/ftp/Specs/html-info/33203.htm. March 2004

Appendix 1: Acronyms and abbreviations

AH	Authentication Header
API	Application Program Interface
AVP	Audio Video Profile
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DNS MX	DNS Mail eXchange record
DNS SRV	DNS resource record for specifying the location of services
CS	Crypto Session
ESP	Encapsulated Security Payload
FW	Fire Wall
GUI	Graphical User Interface
HMAC	Keyed-Hashing for Message Authentication
IKE	Internet Key Exchange
IP	Internet Protocol
IPSEC	IP Security Protocol
KINK	Kerberized Internet Negotiation of Keys
MAC	Message Authentication Code
MIKEY	Multimedia Internet KEYing
MIME	Multipurpose Internet Mail Extension
MKI	Master Key Identifier
NAT	Network Address Translation
PKI	Public Key Infrastructure
RFC	Request For Comment
RTP	Real-Time Transport Protocol
S/MIME	Secure Multipurpose Internet Mail Extensions
SA	Security Association
SADB	Security Association Database
SAVP	Secure Audio Video Profile
SDP	Session Description Protocol
SHA	Secure Hashing Algorithm
SIP	Session Initiation Protocol
SPD	Security Policy Database
SPI	Security Parameter Index
SRTP	Secure RTP
TCP	Transmission Control Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UI	User Interface
URI	Universal Resource Identifier
VoIP	Voice over IP

Appendix 2: Implementation description

The goal of this implementation was to enable the *minisip* user agent to use ESP as security protocol to protect the media stream instead or in combination with SRTP. The IPSEC implementation used is native Linux available in Linux kernel ≥2.5.54 and an API for the kernel is available in ipsec-tools with the libipsec library and libpfkey.h. Libipsec is an implementation of PF_KEY [RFC 2367].

A2.1: MikeyPayloadSP

Minisip needs a way to exchange IPSEC parameters and this is done with the help of MIKEY. To be able to use MIKEY to transport IPSEC parameters the class MikeyPayloadSP was implemented. SP stands for Security Policy and should not be confused with the IPSEC security policy. MIKEY Security Policy transports parameters for the encryption and authentication algorithms. IPSEC security policy is a traffic filter. The MikeyPayloadSP is defined in [RFC 3830]

The CS-ID information for IPSEC was added in the existing class of MikeyCsIdMap. Some larger additions were also made in the classes keyagreement.cxx, keyagreement_dh.cxx and keyagreement_psk.cxx so that the negotiated key agreements could handle IPSEC.

A2.2: SipMIMEContent

To be able to transport more than a single SDP in the SIP message I needed support for multipart MIME. The *minisip* implementation transports the SDP as a SipMessageContent and my solution was to let the class SipMIMEContent be of type SipMessageContent. The SipMIMEContent class can be of any Content-Type as long as the content type can be of string type. This class is used both as Content-Type multipart/mixed and application/mikey. The multipart/mixed message is created in SipDialogVoip and the application/mikey is created in MsipIpsecAPI.

A2.3: MsipIpsecAPI

The MsipIpsecAPI is the tool for *minisip* to manipulate the IPSEC kernel. In this class the dependencies to the operative system dependent ipsec implementation exists. It is this class that is enabled with the configure flag --enable-ipsec-enable.

There are four major functions to this API:

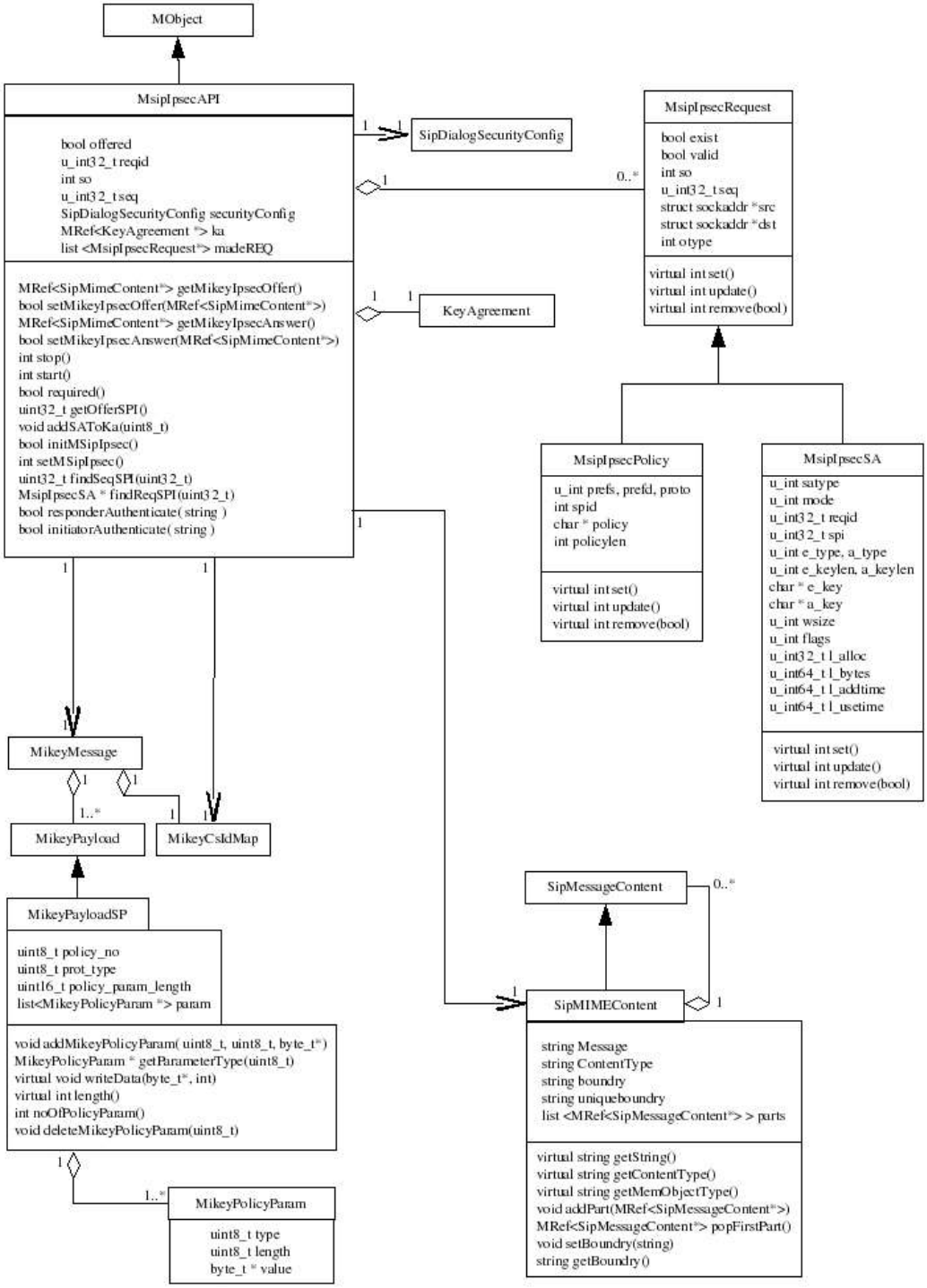
- getMikeyIpsecOffer - that is used when creating the first INVITE, it is called from SipDialogVoip and returns a SipMIMEContent containing a base64 encoded MIKEY message.
- setMikeyIpsecOffer - that is used when receiving the first INVITE, it is called from SipDialogVoip with a SipMIMEContent as argument and returns a bool.
- getMikeyIpsecAnswer - that is used when creating the 200 OK, it is called from SipDialogVoip and returns a SipMIMEContent. This function also sets the IPSEC parameters to the kernel.
- setMikeyIpsecAnswer - that is used when receiving 200 OK, it is called from SipDialogVoip with a SipMIMEContent as argument and returns a bool. This function also sets the IPSEC parameters to the kernel.

This API also keeps track of all requests that has been made to the kernel by a call and whether they are active or not.

Class diagram for the major parts in this implementation is in appendix 2.

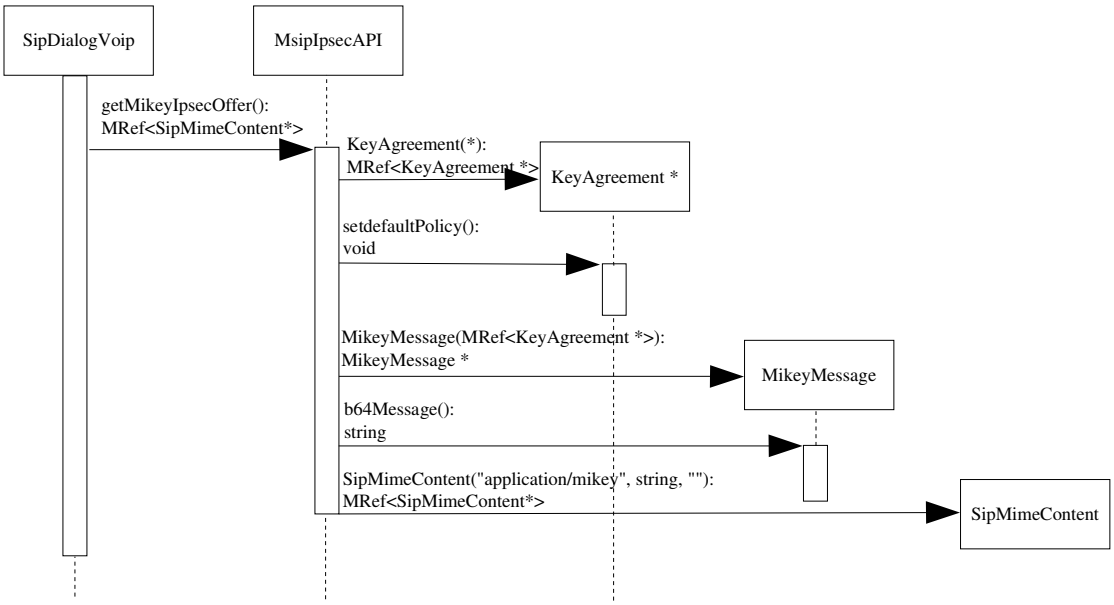
Sequence diagram for major function calls in the MsipIpsecAPI is in appendix 3.

Appendix 3: Class diagram

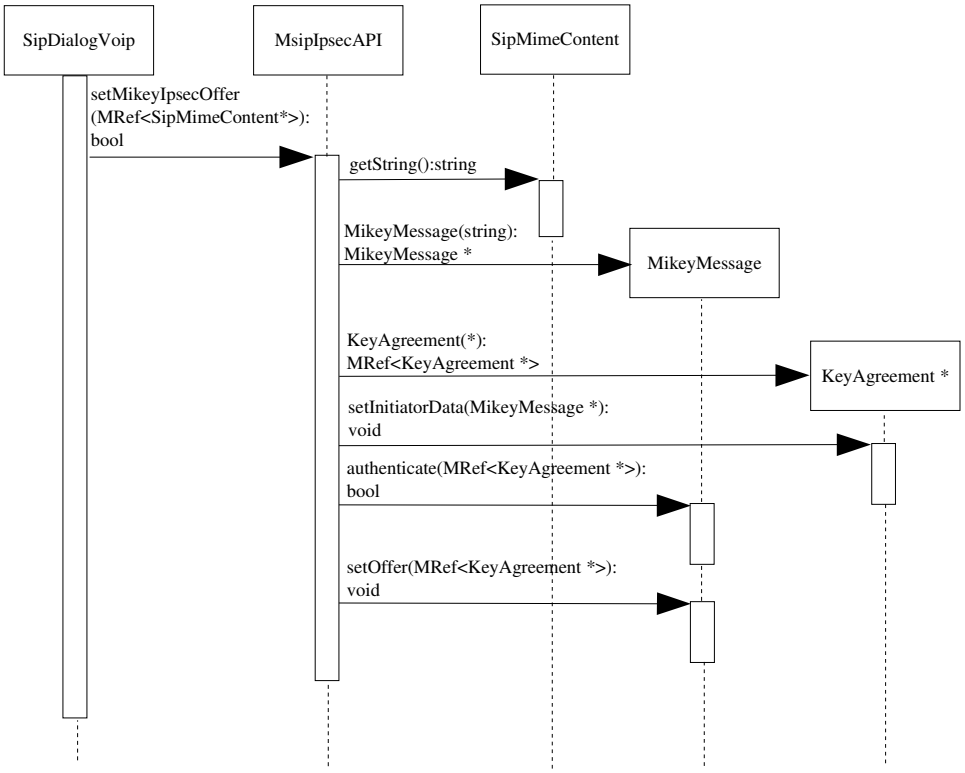


Appendix 4: Sequence diagram

Send INVITE
Major function calls in MsipIpsecAPI

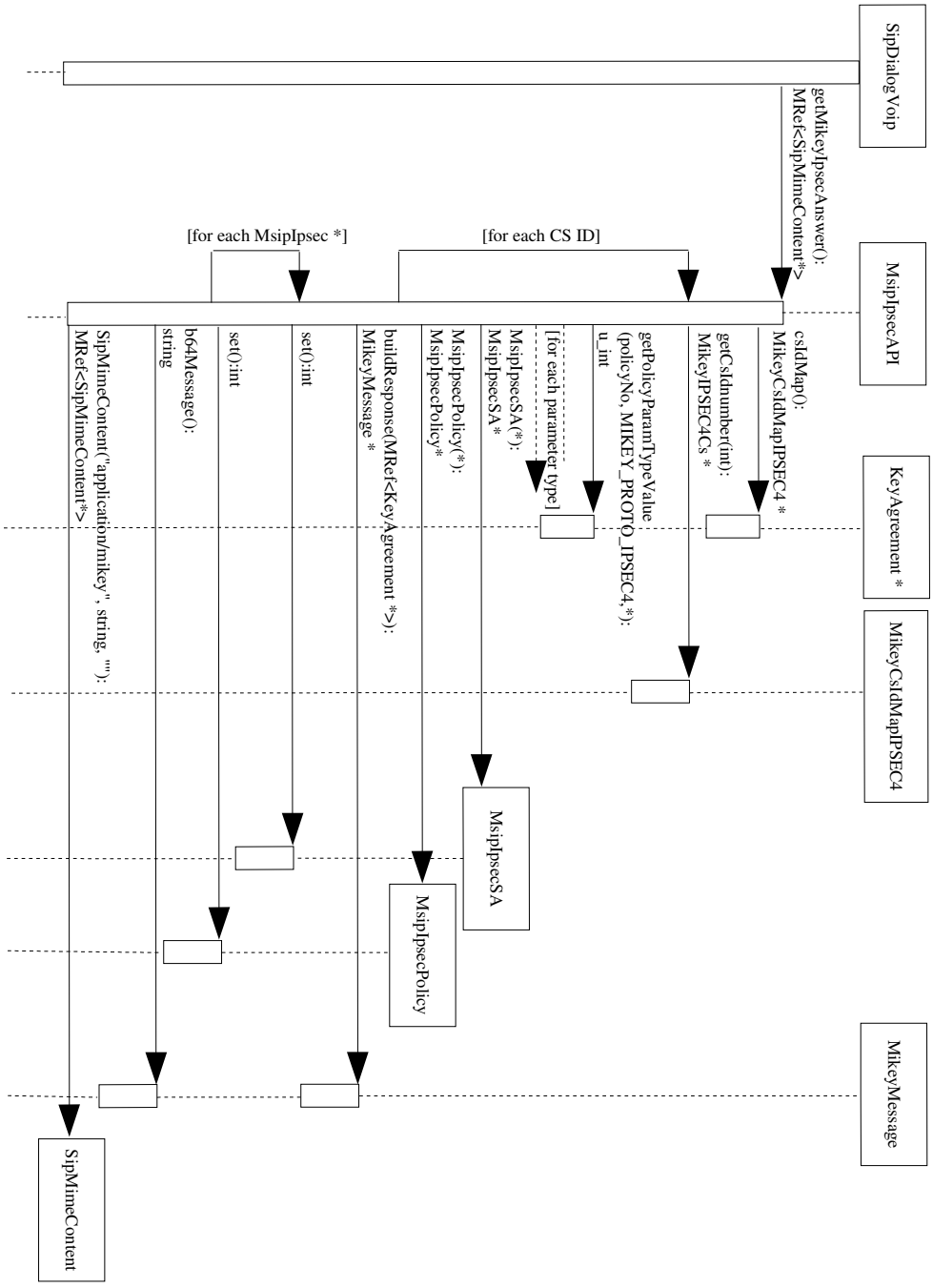


Receive INVITE
Major function calls in MsipIpsecAPI

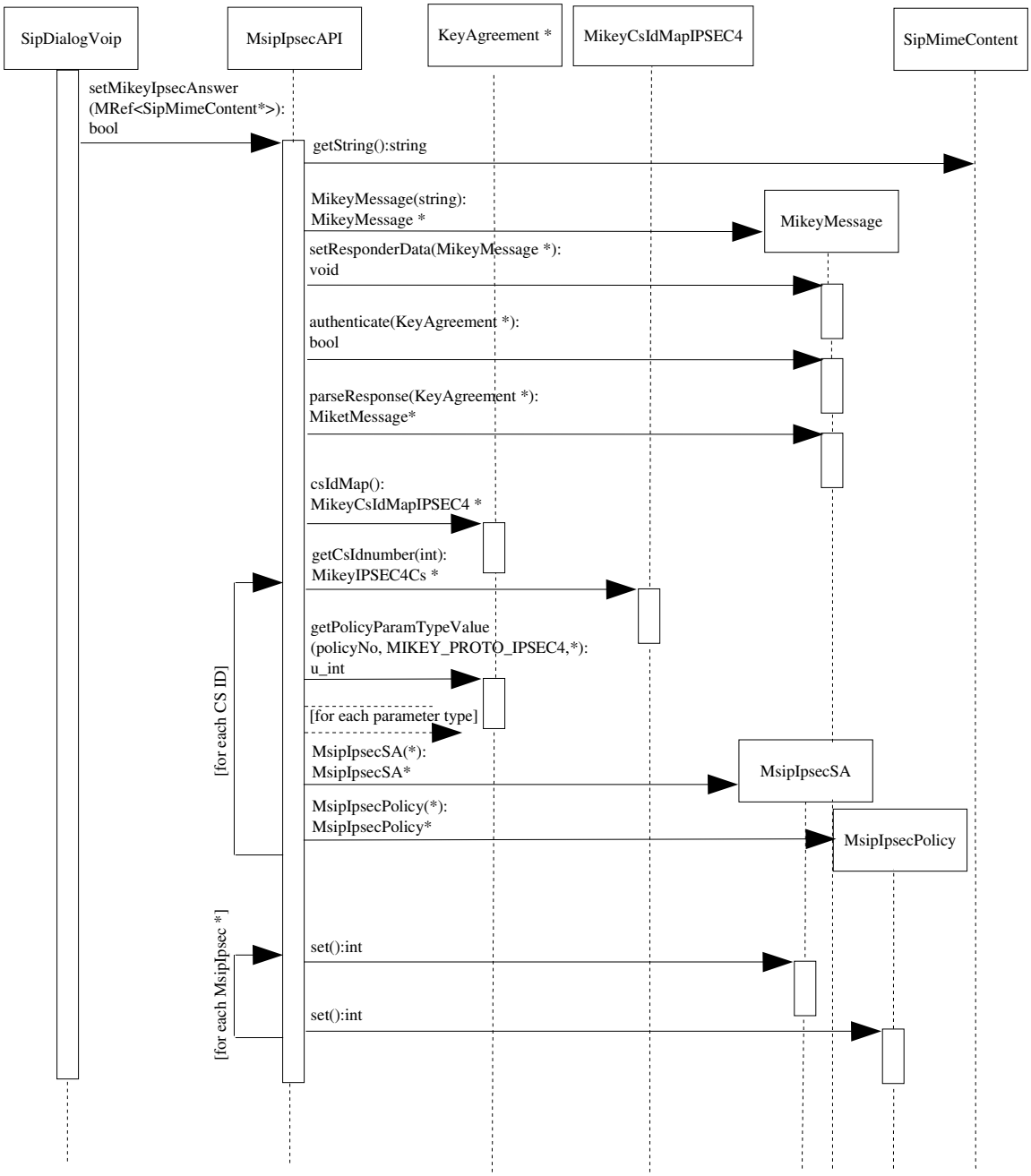


Send 200 OK

Major function calls in MsipIpsecAPI



Receive 200 OK
Major function calls in MsipIpsecAPI



Appendix 5: Measurement raw data

X1-X7 and Y1-Y5 are measured in seconds.

d1-d7 are measured in milliseconds.

A5.1: No security

CLEAR	1	2	3	4	5	6	7	8
X1 "call"	67.174072	45.671575	59.198332	48.448784	90.715679	12.054068	14.502591	55.408725
X2 "SINV"	67.179150	45.675966	59.202673	48.453124	90.720671	12.059087	14.507031	55.413123
X3 "Rring"	67.197477	45.693996	59.220818	48.470835	90.743390	12.076584	14.525736	55.430214
X4 "R200"	77.053891	53.186949	68.430198	56.092216	99.887471	27.136220	23.107176	61.817929
X5 "SACK"	77.056877	53.189922	68.433172	56.095199	99.890445	27.139200	23.110154	61.820907
X6 "setAns"	77.058028	53.191017	68.434272	56.096297	99.891592	27.140295	23.111252	61.822013
Y1 "RINV"	68.783010	47.302987	60.852805	50.127904	92.419027	14.059343	16.542330	57.469966
Y2 "Sring"	68.791015	47.310925	60.860970	50.135884	92.426878	14.067291	16.550282	57.477949
Y3 "Ans"	78.646481	54.802786	70.069523	57.756156	1.568838	29.126453	25.130789	63.864775
Y4 "S200"	78.648670	54.805154	70.071719	57.758356	1.572212	29.128650	25.132984	63.866974
Y5 "RACK"	78.658767	54.815095	70.081793	57.768355	1.582258	29.138710	25.143041	63.876959
d1	5.1	4.4	4.3	4.3	5.0	5.0	4.4	4.4
d2	23.4	22.4	22.5	22.1	27.7	22.5	23.1	21.5
d3	8.0	7.9	8.2	8.0	7.9	7.9	8.0	8.0
d4	2.2	2.4	2.2	2.2	3.4	2.2	2.2	2.2
d5	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
d6	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1
d7	12.3	12.3	12.3	12.2	13.4	12.3	12.3	12.2

	Average	Stddev	Median
d1	4.6	0.3	4.4
d2	23.2	1.9	22.5
d3	8.0	0.1	8.0
d4	2.4	0.4	2.2
d5	3.0	0.0	3.0
d6	4.1	0.0	4.1
d7	12.4	0.4	12.3

A5.2: SRTP MIKEY with pre-shared secret

SRTP (psk)	1	2	3	4	5	6	7	8
X1 "call"	56.620254	89.213770	88.134688	60.639421	77.547368	33.035480	60.632251	84.090223
X2 "SINV"	56.627909	89.220756	88.142306	60.647037	77.555237	33.043157	60.639230	84.097769
X3 "Rring"	56.649865	89.242573	88.162485	60.668269	77.580023	33.063467	60.661275	84.117734
X4 "R200"	63.380726	97.016946	92.825405	67.993864	82.020169	39.103901	66.186058	90.889502
X5 "SACK"	63.383906	97.020083	92.828557	67.996998	82.023316	39.107039	66.189211	90.892660
X6 "setAns"	63.385485	97.021673	92.830158	67.998642	82.024910	39.108658	66.190853	90.894304
Y1 "RINV"	59.187578	91.459618	90.399536	62.937455	80.021103	35.520774	63.165365	86.637174
Y2 "Sring"	59.197539	91.469598	90.409609	62.947470	80.030890	35.530660	63.175285	86.647085
Y3 "Ans"	65.925413	99.241095	95.071107	70.271750	84.469086	41.568010	68.698242	93.417080
Y4 "S200"	65.929215	99.244888	95.073702	70.274340	84.471689	41.571880	68.701297	93.419675
Y5 "RACK"	65.939729	99.255419	95.083754	70.284483	84.482127	41.582337	68.711403	93.430183
d1	7.7	7.0	7.6	7.6	7.9	7.7	7.0	7.5
d2	29.6	28.8	27.8	28.8	32.7	28.0	29.0	27.5
d3	10.0	10.0	10.1	10.0	9.8	9.9	9.9	9.9
d4	3.8	3.8	2.6	2.6	2.6	3.9	3.1	2.6
d5	3.2	3.1	3.2	3.1	3.1	3.1	3.2	3.2
d6	4.8	4.7	4.8	4.8	4.7	4.8	4.8	4.8
d7	14.3	14.3	12.6	12.7	13.0	14.3	13.2	13.1

	Average	Stddev	Median
d1	7.5	0.3	7.6
d2	29.0	1.6	28.8
d3	9.9	0.1	9.9
d4	3.1	0.6	2.8
d5	3.1	0.0	3.1
d6	4.8	0.0	4.8
d7	13.5	0.7	13.1

A5.3: ESP MIKEY with pre-shared secret

IPSEC (psk)	1	2	3	4	5	6	7	8
X1 "call"	78.225137	82.179065	53.273646	10.963880	48.159130	59.626558	70.462927	57.375338
X2 "SINV"	78.232538	82.186992	53.281625	10.971926	48.16705	59.63461	70.47095	57.382728
X3 "Rring"	78.256693	82.208278	53.303399	10.994076	48.209736	59.655841	70.492614	57.403310
X4 "R200"	87.203544	89.190675	62.984959	18.464370	67.135215	69.326041	77.154430	63.990625
X5 "SACK"	87.206749	89.193874	62.988181	18.467545	67.138405	69.329223	77.157667	63.993787
X6 "setAns"	87.870939	89.860223	63.652139	19.147735	68.045808	69.995785	77.850713	64.658153
Y1 "RINV"	75.450065	79.433697	50.553949	12.447998	49.216863	61.162934	71.954436	58.437458
Y2 "Sring"	75.459730	79.443561	50.563696	12.457922	49.226495	61.172790	71.964248	58.447219
Y3 "Ans"	83.735223	85.754838	59.573207	19.260335	67.318246	70.172652	77.959551	64.361929
Y4 "S200"	84.407087	86.426483	60.245893	19.928618	68.153087	70.844625	78.625988	65.034795
Y5 "RACK"	84.418106	86.437542	60.256895	19.939669	68.164179	70.855658	78.636959	65.045848
d1	7.4	7.9	8.0	8.0	7.9	8.0	8.0	7.4
d2	31.6	29.2	29.8	30.2	50.6	29.3	29.7	28.0
d3	9.7	9.9	9.7	9.9	9.6	9.9	9.8	9.8
d4	671.9	671.6	672.7	668.3	834.8	672.0	666.4	672.9
d5	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
d6	667.4	669.5	667.2	683.4	910.6	669.7	696.3	667.5
d7	682.9	682.7	683.7	679.3	845.9	683.0	677.4	683.9

	Average	Stddev	Median
d1	7.8	0.3	8.0
d2	32.3	7.5	29.7
d3	9.8	0.1	9.8
d4	691.3	58.0	671.9
d5	3.2	0.0	3.2
d6	704.0	84.1	669.6
d7	702.4	58.1	682.9

d4 and d6 includes the setting of SA to the kernel, 2 calls to the function pfkey_send_add from libipsec in both d4 and d6. Each function call takes about 330mS.

Appendix 6: Original thesis description

This is the original thesis description from IMIT/TsLab KTH

Alternatives to MIKEY/SRTP to secure VoIP (IKE/IPSEC and others)

Goal:

1. Investigate what alternative keying and secure transport protocols that exist to MIKEY/SRTP for secure VoIP and how those can be integrated into a SIP call setup (e.g., IKE/IPSec)
2. Implement/demonstrate a suitable alternative into minisip

Background:

In order to demonstrate secure VoIP services we have release an open source SIP user agent (minisip) supporting end-to-end authentication and media protection. To protect the media streams minisip currently uses Secure RTP (SRTP) and as authenticated keying protocol minisip uses Multimedia Internet Keying (MIKEY), however, this master thesis would concern alternatives to MIKEY/SRTP. See www.minisip.org for more information on minisip. As keying mechanism one could consider e.g. IKE, TLS or KINK. One problem that arises would be how to exchange, e.g., IKE messages, between the two end-nodes. MIKEY messages are carried inside SIP messages (as SDP attributes in e.g. the SIP INVITE message). For IKE and TLS an interesting alternative would be to use SIP only for exchanging basic connectivity information (IP address, port numbers etc.), possibly via a SIP OPTIONS message, and then run IKE/TLS as usual. To secure the media stream one could still use SRTP (possibly using SDP attributes as in an expired suggestion by Baugher (draft-baugher-mmusic-sdpmediasec-00.txt)), but IPSec is an interesting alternative, in particular if IKE is used as keying protocol. Another approach would be to look at security work for IKE/IPSec and SIP that has been done within 3GPP.