

# Secure Text in SIP Based VoIP

JOHAN KULTTI

**MASTER OF SCIENCE PROGRAMME**  
**Computer Science**

Luleå University of Technology  
Department of Computer Science and Electrical Engineering  
Division of Software Engineering



# Secure Text in SIP Based VoIP

Master of Science Thesis  
Luleå, May 2005

Johan Kultti

Luleå University of Technology

Supervisor: Andreas Piirimets  
Examiner: Peter Parnes

### **Abstract**

Historically, as new technologies evolve and start to become viable business tools there has been a tendency for developers to consider the addition of security features a performance decreasing nuisance. However, with security becoming an increasingly important issue in modern day computer environments, it is becoming vital to consider how to protect a system before incorporating it in daily business operations. Due to the increasing interest in computer based media communication, for example VoIP applications, there exists a need for security solutions which make these technologies reliable enough to carry important information. This thesis examines how to secure a SIP session and associated media streams, in particular considering media streams carrying 'RTP Payload for Text Conversation' (RFC 2793) data. The conclusion proposes a layered security where additional features can be developed and added without impacting existing security measures. The project serves as another step in a collaborative effort to promote mainstreaming and functional enhancements towards telecommunications access for all. Earlier efforts have produced a product to provide text conversation in real-time applications, the purpose of which is to provide the hard of hearing with communication opportunities with those not proficient in sign language. Securing this type of application is an important step in making it an attractive means of communication.

## Preface

This document was created as a part of the master degree project of Johan Kultti at the Master of Science Programme in Software Engineering at Luleå University of Technology. The work was carried out at Omnitor AB during the fall of 2004 and spring of 2005.

I would like to take the opportunity to express my sincere thanks to:

- My supervisor Andreas Piirimets, and Erik Zetterström of Omnitor AB for their invaluable help and feedback during the course of this project.
- Peter Parnes, Luleå University of Technology, who, despite his many commitments, agreed to take on the role of examiner for this project.
- Gunnar Hellström, Omnitor AB, for his interest and advice.
- My family for the support they have always given me in everything I have ever done.

This work was partially funded by the National Institute on Disability and Rehabilitation Research, US Dept of Education under Grant H133E040013 as part of a co-operation between the Telecommunication Access Rehabilitation Engineering Research Center of the University of Wisconsin -Trace Center joint with Gallaudet University, and Omnitor. The goal was to promote mainstreaming and functional enhancements toward telecommunications access for all. The opinions herein are those of the authors and not necessarily those of the funding agencies.

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	Project Background .....	7
1.2	Project Objectives .....	7
1.3	Terminology .....	8
<b>I</b>	<b>SIPcon1 Current Practices .....</b>	<b>11</b>
<b>2</b>	<b>SIPcon1 Software Protocols .....</b>	<b>12</b>
2.1	Session Establishment .....	12
2.2	Media Negotiation .....	12
2.3	Media Transfer .....	12
2.4	Media Content .....	12
<b>3</b>	<b>SIPcon1 Protocols and Standards Descriptions .....</b>	<b>14</b>
3.1	SIP .....	14
3.2	SDP .....	16
3.3	RTP .....	17
3.4	Media Content – RFC2793 .....	18
<b>II</b>	<b>Security Threats and Goals .....</b>	<b>20</b>
<b>4</b>	<b>Threat Model for SIP and Real-time Media Transfer .....</b>	<b>21</b>
4.1	Attackers .....	21
4.2	Target Resources .....	21
4.2.1	<i>End Systems</i> .....	21
4.2.2	<i>Application Users</i> .....	22
4.2.3	<i>SIP Registrar and Registration</i> .....	22
4.2.4	<i>Location Service</i> .....	22
4.2.5	<i>Proxy Servers</i> .....	22
4.2.6	<i>Network Infrastructure</i> .....	23
4.2.7	<i>SIP Messages</i> .....	23
4.2.8	<i>Media Transfer</i> .....	23
<b>5</b>	<b>Security Goals .....</b>	<b>25</b>
5.1	Registration .....	25
5.1.1	<i>SIP Registration Authentication and Integrity</i> .....	26
5.1.2	<i>SIP Registration Confidentiality</i> .....	27
5.2	SIP Communication .....	27
5.2.1	<i>Session Message Authenticity and Integrity</i> .....	27
5.2.2	<i>Session Message Confidentiality</i> .....	28
5.3	Media Transfer .....	29
5.3.1	<i>Media Authenticity and Integrity</i> .....	29
5.3.2	<i>Media Confidentiality</i> .....	30

---

<b>III Security Solutions</b>	<b>31</b>
<b>6 Encryption Protocols</b>	<b>32</b>
6.1 IPsec: Network Layer Encryption	32
6.2 TLS: Transport Layer Encryption	33
6.3 SRTP: Application Layer Encryption	33
<b>7 Media Encryption</b>	<b>37</b>
7.1 IPsec: Network Layer Encryption	37
7.2 TLS: Transport Layer Encryption	39
7.3 SRTP: Application Layer Security	39
<b>8 Key Agreement</b>	<b>40</b>
8.1 Standard Key Agreement protocol	40
8.1.1 <i>Pre-Shared Key Agreement</i>	40
8.1.2 <i>Public-key Agreement</i>	41
8.1.3 <i>Diffie-Hellman Key Agreement</i>	41
8.2 Key Management Protocols	43
8.2.1 <i>ISAKMP</i>	43
8.2.2 <i>IKE</i>	43
8.2.3 <i>MIKEY</i>	45
<b>9 Media Security Negotiation</b>	<b>47</b>
9.1 Independent Key Management	47
9.2 The SDP Encryption Key Attribute	47
9.3 SDP Security Negotiation Protocols	47
9.3.1 <i>SDP Security Descriptions for Media Streams</i>	48
9.3.2 <i>Key Management Extensions for SDP</i>	49
<b>10 SIP Encryption</b>	<b>52</b>
10.1 Hop-by-hop Security	52
10.1.1 <i>SIPS using TLS</i>	52
10.1.2 <i>SIP with IPsec</i>	55
10.2 End-to-end Security	56
10.2.1 <i>S/MIME</i>	57
10.2.2 <i>Public Key Infrastructures</i>	58
10.2.3 <i>Enhancements for Authenticated Identity Management in SIP</i>	59
<b>11 SIP Security Negotiation</b>	<b>60</b>
11.1 Default Session Security	60
11.2 Security Mechanism Agreement for SIP	60
<b>IV Results and Conclusions</b>	<b>61</b>
<b>12 SRTP Implementation</b>	<b>62</b>
<b>13 Security Scenarios</b>	<b>64</b>
13.1 Light Security	65
13.2 Medium Security	66
13.3 Heavy Security	66

---

**14 Conclusions ..... 68**  
**15 References and Bibliography ..... 69**

**Appendix A - Acronyms and Abbreviations**

# 1 Introduction

## 1.1 Project Background

Omnitor is a company dedicated to developing information technology suitable for people with disabilities. Communication being an important part of today's society, it is vital to make new communication possibilities available to as many people as possible. By providing software and tools that alleviate some of the difficulties associated with a particular disability new technology becomes available for people who would otherwise receive yet another disability by not being able to use the new technology. The telephone is an example of an important technological advance that has not been able to provide equal opportunity all users.

The 'RTP Payload for Text Conversation' specification [16] has been developed by Gunnar Hellström, Omnitor AB, to provide a standard for payload format in real-time text conversation. The use of this technology enables two participants to communicate using real-time text. By extension, the versatility inherent in computer-based software allows the implementation of this standard to provide text communication without additional devices when used in an IP-telephony context. In other words, without having to use intermediary translation or extra equipment not available to everyone someone without knowledge of sign language can communicate with someone dependent upon it.

Omnitor has created an application called SIPcon1 to demonstrate the use of RFC 2793 [16] based text conversation. Since the standard does not provide specifics on how to provide security mechanisms for the communication all data is sent in plaintext. Because communication between two compatible computer applications with access internet implies the use of a packet switched network any information exchanged should be considered more exposed than information exchanged over a circuit switched network (used in conventional telephony). For this kind of application, security mechanisms to ensure the privacy of the communication should be considered a vital part of making the application attractive to users.

## 1.2 Project Objectives

RFC 2793 provides no information on how to provide security mechanisms for the text communication. The purpose of this document is to give an overview of, and possible solutions to the problems associated with providing security mechanisms to an application using said format to provide text communication. Since the SIPcon1 reference implementation uses the 'Session Initiation Protocol' [37] (SIP) for session initiation, thus providing means for interoperability with SIP compliant applications, the focus herein lies on how to implement security measures for an application using SIP to establish the session.

Simplifying the security problem the process of providing security for the communication of data between two arbitrary users consists of two basic problems:

- Media encryption
- Secure Key agreement

The method of encryption dictates how encryption, decryption and hashing of the media is to be performed to provide the security features needed by the application. These features provide the most important benefits of adding security mechanisms to the application and are explained in Chapter 5, Security Goals. We seek an encryption method suitable for the transfer of RTP text conversation payload data, as defined in RFC 2793, such that parties without authorized access are unable to participate in or eavesdrop on the communication within a reasonable timeframe.

Perhaps the more complex of the two problems, agreement on encryption parameters is needed because methods of encryption generally rely on common parameters and shared secrets to exclude unauthorized access. The complexity is brought on by the paradoxical need for the secret exchange of these parameters without first sharing a secret to encrypt the exchange with. Because of the complexity of this problem many security implementations rely on ad-hoc solutions, such as pre-shared secrets, which do not scale well to larger scenarios. In this document the process of exchanging the necessary cryptographic information will be referred to as key agreement, even though the process may in fact involve the exchange of more than just keys such as deciding upon method of encryption. The goal for this key agreement is to provide the keying material needed by the method of encryption in a secure manner so that the exchange does not compromise the security of the subsequent encryption. This implies that the process of key agreement itself needs security mechanisms similar to those needed for the encryption of the actual media stream.

In addition to providing methods of encryption and key agreement any suggestions made will be based on additional requirements. The requirements can be based on Quality of Service (QoS) issues, interoperability, extendibility or requirements brought on by the vulnerability of a certain mechanism.

The security goals of an application can vary greatly depending on the use and users of that application. Thus, a text communication tool might need varying degrees of security depending on communication content and the users involved. Considering there are no “one size fits all” security mechanisms the conclusion will reflect this by suggesting different setups for different scenarios.

### 1.3 Terminology

As with any technical subject the area of encryption involves a number of expressions which might not be familiar to those not knowledgeable in that area. Some expressions used throughout the report will be briefly explained so as to clarify their meaning in the context of this project. Many of these explanations are taken directly from a protocol made obsolete [5] that nevertheless amply explains some of the terminology that will be used.

#### **Authentication**

*“The property of knowing that the data received is the same as the data that was sent and that the claimed sender is in fact the actual sender” [5].*

#### **Confidentiality**

*“The property of communicating such that the intended recipients know what was being sent but unintended parties cannot determine what was sent” [5].*

**Decryption**

Reversing the effects of encryption.

**Denial of Service**

An event in which some or all of an application's functionality is prevented.

**Encryption**

*"A mechanism commonly used to provide confidentiality"* [5].

**Hashing**

Creating a hash, or message digest, substantially smaller than the segment of data used as input in such a way that any other data segment processed in the same way is extremely unlikely to produce the same hash.

**Integrity**

*"The property of ensuring that data is transmitted from source to destination without undetected alteration"* [5].

**Man-in-the-middle Attack**

The process of launching an attack that directs the traffic of two communicating parties through an entity controlled by the attacker.

**Message Spoofing**

Creating a faked message with the intention of trying to pass it off as a legitimate message.

**Perfect Forward Secrecy**

The attribute for a key exchange mechanism where an exposed key does not jeopardize future transactions.

**Quality of Service**

A term used to describe the performance properties of a network service.

**Replay Attack**

The re-injection of a previously sent, legitimate packet into the network.

**Security Association**

*"The set of security information relating to a given network connection or set of connections"* [5].

**Social Engineering**

A popular term used to describe the manipulation of people for the purpose of attaining information harmful to the security of a system.

**Traffic Analysis**

*"The analysis of network traffic flow for the purpose of deducing information that is useful to an adversary"* [5].

**User Agent**

A logical entity that can act as both User Agent Client and User Agent Server [37].

**User Agent Client**

A logical entity that creates and sends a new SIP request. The User Agent Client role lasts for the duration of the transaction [37].

**User Agent Server**

A logical entity that generates a response to a SIP request, either accepting, rejecting or redirecting the request. The User Agent Server role lasts for the duration of the transaction [37].

## **I SIPcon1 Current Practices**

## 2 SIPcon1 Software Protocols

Securing the text is important for any SIP based application using real-time text conversation as defined by RFC 2793. To represent this generic application, in many instances, Omnitor's SIPcon1 application will be used as an example.

SIPcon1 is a Java reference implementation of the IETF 'RTP Payload for Text Conversation' standard [16], developed by Omnitor, the purpose of which is to convey knowledge of real-time text transfer. The application uses the 'Session Initiation Protocol' [37] (SIP), the 'Session Description Protocol' [13] (SDP) and the 'Real-time Transport Protocol' [38] (RTP) to achieve the communication similar to the manner in which VoIP calls, conference tool sessions and whiteboard application sessions are initiated and run.

### 2.1 Session Establishment

Programs such as SIPcon1 need a way to establish a connection to another client if they are to communicate. With different programs wanting to communicate with each other in different ways a common method of establishing these connections is needed. SIPcon1 uses a protocol called the 'Session Initiation Protocol' [37] (SIP) to establish a session between two clients. The SIP protocol was conceived for the purpose of creating, modifying and terminating sessions with one or more participants and is used extensively in the establishment of VoIP sessions.

### 2.2 Media Negotiation

As there are many types of media, and every type of media has several ways of encoding the information, there needs to be a way of negotiating what types of media and what types of encoding are to be used in the session. There also needs to be agreement on what ports to use for the media communication. The 'Session Description Protocol' [13] (SDP) provides a way for SIP messaging to negotiate media types, port addresses and encoding by providing a standardised format for session description.

### 2.3 Media Transfer

Since SIPcon1 is meant to distribute real-time media it needs a transport protocol made for speed. By using an implementation of the 'Real-time Transport Protocol' [38] (RTP) this is achieved. RTP is an UDP based transport protocol developed for real-time media streaming and does not provide guaranteed delivery of sent packets.

### 2.4 Media Content

SIPcon1 is used to send audio, video and textual data from one client to another. Since the purpose of this thesis is to secure the textual data this media form will be the main focus on

which all assumptions are based, although extensibility to other forms of media will be an important consideration wherever appropriate. The text sent in SIPcon1 follows the 'RTP Payload for Text Conversation' [16] (RFC 2793) standard. This format has been specifically developed as a format suitable for straightforward real time text conversation and the SIPcon1 software has been constructed to demonstrate the use of this standard. Because of the nature of unreliable delivery protocols such as the User Datagram Protocol (UDP) (which RTP is based on) RTP is not ideally suited for delivery of textual data. The payload format of RFC 2793 is used to make the use of RTP more suited for this purpose by advocating the use of redundancy to counter the effects of packet loss.

### 3 SIPcon1 Protocols and Standards Descriptions

To demonstrate the use of ‘RTP payloads for text conversation’ [16] the SIPcon1 application uses a number of protocols as explained in the previous chapter. Extending the functionality of such an application to include security mechanisms is done in consideration of the protocols already in use. The protocols described in this chapter are implemented to give the application its current functionality.

#### 3.1 SIP

The ‘Session Initiation Protocol’ [37] (SIP) is an application-layer protocol developed to handle the initiation, modification and termination of sessions. A couple of the problems associated with initiating a media session are locating participants, and negotiating session specific information. SIP defines how to locate and contact a desired participant while carrying session specific information in the “Invite” message body and its corresponding answer message. SIP relies on another protocol (SDP [13]) for the creation and interpretation of the message bodies which carry the media information. By only defining how session negotiation is communicated SIP can be used in a multitude of application environments independently of what type of session is being established or the underlying transport protocols used for the media transfer.

SIP enables the creation of an infrastructure that can relay messages from the call initiator to the call recipient even if the actual location of the recipient is unknown [37]. This is accomplished by having each user agent (an application which can both send and receive data) register its Uniform Resource Identifier (URI), a kind of network address, with a domain registrar entity which stores the user’s location information in a location service. Whenever a user agent sends a SIP message it is directed at a proxy server, which may or may not be the same entity as the registrar, entrusted to convey the message to the next so called hop. The proxy server, part of a network of such servers, directs the message to the next proxy along the intended route unless the intended recipient is registered at the same domain as the proxy in which case the message is forwarded to the recipient. After the initial session parameters have been decided subsequent SIP messages may be sent end-to-end, as the receiver location is now known, or along the same route as the initial messages.

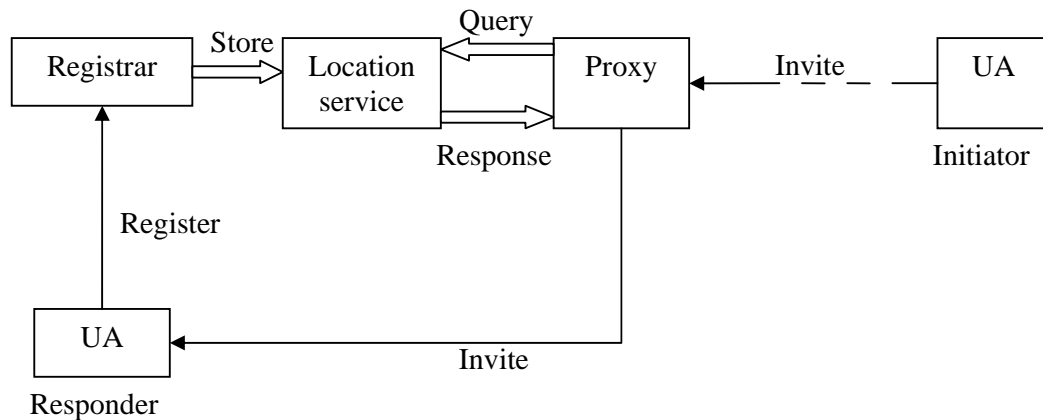


Figure 3.1: Routing of a SIP Invite message

Figure 3.1 shows how an invite message is passed to its intended recipient. The Responder must first have registered with his registrar to provide routing information. The broken line of the initiators invite message indicates that the invite of a user agent outside of the responder's domain would first have passed through the SIP infrastructure of the initiators domain, querying at the outbound proxy for the address of the recipients inbound proxy. Note that since the registrar, the outbound proxy and inbound proxy are logical roles, not physical, [37] they may or may not be represented by different entities.

By simplifying the domain infrastructure to a single entity the SIP call establishment process between two registered users on separate domains proceeds as indicated by Figure 3.2, with messages numbered in sequential order. After the call initiation any negotiated media starts flowing directly between the user agents.

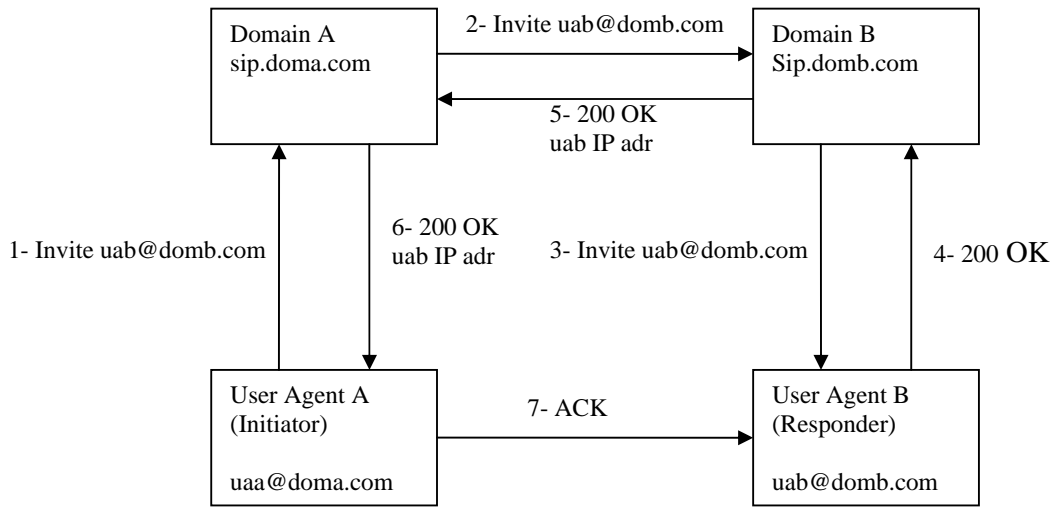


Figure 3.2: Overview of SIP session establishment

### 3.2 SDP

The ‘Session Description Protocol’ [13] (SDP) defines syntax and semantics for negotiating media streams or, more generally, session contents. SDP is a textual protocol that allows a session initiator to signal several suggested streams and configurations with the knowledge that the return message from the recipient will contain the first supported configuration.

The means with which media is to be sent need to be negotiated before any communication can be initiated. There needs to be agreement on what type of media to send and with what protocol, and to which port, to send them. The service provided by SDP caters to all these requirements.

When two applications exchange media the data has to be compressed and structured in some way. Since compression/decompression algorithms, known as CODECs, that do this compression is not interoperable, SIP session participants need to know what CODECs to use (and at what sample rates) and be sure both systems share the required CODECs. The problem of codec agreement is somewhat similar to that of encryption parameter agreement and, as later solutions will show, this makes an extension to SDP a good option for conveying cryptographic information.

```
v=0
o=Pontus_WX3 1 1 IN IP4 192.168.1.30
s=Omnitor_SipController
c=IN IP4 192.168.1.30
t=0 0
m=audio 12000 RTP/AVP 5 4 3
a=rtpmap:5 DVI4/8000
a=rtpmap:4 G723/8000
a=rtpmap:3 GSM/8000
m=text 10000 RTP/AVP 99 98
a=rtpmap:98 T140/1000
a=rtpmap:99 RED/1000
a=fmtp:99 98/98
```

Example 3.1: A SIPcon1 offer SDP description example

In Example 3.1, demonstrating what a SIPcon1 session SDP description might look like, each m field signifies one media, describing type of media, destination port and transport protocol. The first media field suggests audio communication on port 12000 using RTP and the CODECs mapped to 5, 4, and 3. The media announcement fields below each m line suggest CODECs and sample rates in order of preference. A responder to this message who, of the suggested CODECs in the audio stream media description field, only supported GSM would answer using that media announcement. If that same user does not support a media at all, say for example the text communication, the port in the response is set to 0. The answer would look like that of Example 3.2.

```
v=0
o=Pontus_WX3 1 1 IN IP4 192.168.1.30
s=Omnitor_SipController
c=IN IP4 192.168.1.30
t=0 0
m=audio 12000 RTP/AVP 3
a=rtpmap:3 GSM/8000
m=text 0 RTP/AVP 99 98
```

Example 3.2: Possible response to the offer in Example 3.1

### 3.3 RTP

When SIP and SDP has been used to set up session parameters all that remains, except modifying or ending the session, is the actual encoding and transfer of the media. The SIPcon1 application uses the Java Media Framework (JMF) for the sampling of audio and video feeds because functions to sample and distribute media streams are readily available in JMF. However, the transmission of the sampled data is not handled by JMF but by Omnitor's

RTP implementation. As the purpose of this thesis is to conceive a way of securing the text media the use of RTP for transmitting text is the main focus of this section.

The ‘Real-time Transport Protocol’ [38] (RTP) is based on the ‘User Datagram Protocol’, UDP, for the transportation of data and hence does not guarantee delivery of sent data. RTP is mainly meant to send large quantities of data, where guaranteed delivery is less important than congestion of the network or where re-sent, and thus late, packets would be of no use (or a combination of both reasons).

Text transmission falls way short of the ideal use of RTP when considering the form in which this media is created and presented. Transmission of text created on the fly by user input does not require enough bandwidth to motivate the choice of RTP, and because missing text packets could cause large discrepancies between receiver interpretation and sender intent text transmission requires that lost packets be recovered.

The use of RTP as a transport protocol for the text, other than as a basis for experimentation, can be motivated by the use of RTP for the transportation of the other media streams. By using the same transport protocol changes or additions made to the application are easily extended to apply to all transmitted media.

### 3.4 Media Content – RFC2793

Because RTP uses UDP for the transportation of data the overlying application must handle any retransmission, packetization and reassembly. The importance of reliable delivery for the interpretation of text media sets requirements for how text data needs to be handled.

Gunnar Hellström, Omnitor AB, has developed the ‘RTP Payload for Text Conversation’ [16] (RFC 2793) standard in an attempt to standardize real-time text conversation. The SIPcon1 software has been constructed by Omnitor AB as a reference implementation to demonstrate the use of real-time text conversation. Text conversation contents follow the ITU-T T.140 [15] recommendation for textual conversational communication between two participants. Packet loss issues sprung from the use of UDP with textual conversation are solved, or mitigated, using the “RTP Payload for Redundant Audio Data” [32] standard. This standard, originally meant to be used for the distribution of audio data, introduces the use of redundant data to recover lost information. This is possible, providing the number of consecutive packets lost is lower than the number of redundant blocks in each packet, by including the data of some previously sent packets in every new packet. The newest draft of RFC 2793bis [17], meant to eventually replace the current version of RFC 2793, recommends the use of two redundant blocks with every packet ensuring that all text data is delivered as long as one of three consecutive packets arrives at its intended destination. Packet loss of a value of four consecutive packets would in this case entail loss of the data contained within the oldest redundant packet and subsequent lost packets would each create loss of one packet of data until a packet safely reaches its destination. By using redundancy in this way a buffer of sorts is maintained that allows for two consecutive packets to be lost without loss of data at the receiving end.

Since text conversation is not a continuous media transfer, specifically there might be pauses between writing a sentence and waiting for a reply, only sending redundant data with new packets could create noticeable loss of characters at the end of a sentence until the next

sentence is started. RFC 2793 therefore advises that an application should create completely redundant packets, with no new information, whenever it detects no new characters have been sent within a certain time interval.

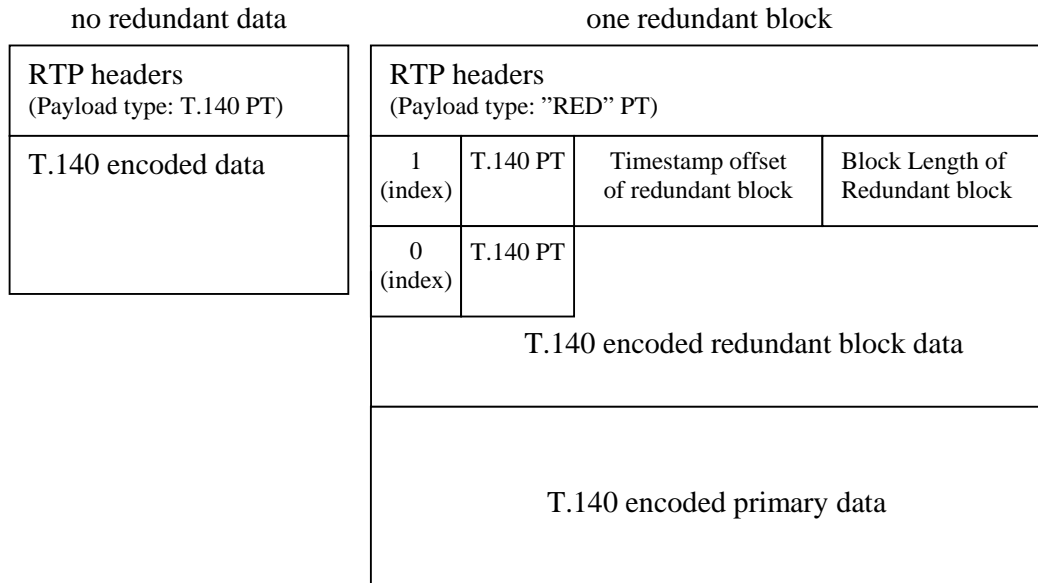


Figure 3.3: T.140 payloads with and without redundant data

Figure 3.3 demonstrates the added fields required for use of redundancy in the T.140 payload. A block index, timestamp offset and block length field as shown in the payload using redundancy is required for every block of redundant data. The block index 0 indicates there are no more redundant blocks in the payload (and hence all remaining data not accounted for by the block length fields must be new primary data).

## **II Security Threats and Goals**

## 4 Threat Model for SIP and Real-time Media Transfer

One important factor when deciding upon a security mechanism is the threat model [7]. Getting a good understanding of the types of threats that exist against a particular application is essential to creating or selecting security mechanisms suitable for that application. The threat model usually takes into account any specific sources of attack, the expected target of attack and the means with which such an attack is carried out.

### 4.1 Attackers

A diverse application such as a teleconferencing system can be subjected to attack by many sources. As such an application is only a means of communication between two parties, any goal an attacker might have depends greatly on who the users are and what information they are transmitting.

A company using a multimedia application for transmission of audio and video data (and possibly other data types such as text, whiteboard data, etc.) might depend on the application for communication between company employees and for online conferencing. An attacker involved in industrial espionage could obtain company secrets, alter information and interrupt daily operation if the application is left vulnerable to attack. An attacker with no interest in company information could still make secret information known to the general public or sabotage the company's ability to communicate effectively.

A person, be it a company employee or someone else, might use the application for purely private communication. An attacker eavesdropping on a private conversation might still get access to insurance information, security codes, medical information or other sensitive information. Even if this is not the case (the user might be aware of the fact that the application is not "safe") the ability for someone to eavesdrop on a private conversation could be considered such a violation of personal integrity that the user would simply not use the application.

### 4.2 Target Resources

Identifying resources which might be targeted by an attacker allows the deployment of security mechanisms which counter threats on those particular resources. It is important to note that not all targets of malicious attack are possible to protect using software measures and good security policies are needed to reduce the risks of these attacks.

#### 4.2.1 End Systems

Making sure the end system of a user is inaccessible to unauthorised access might seem rather obvious, but in practice this is one of the security aspects most commonly forgotten. Many times companies implement strong security mechanisms for their systems and, in feeling their systems are already safe, fail to consider the human aspect of computer security. There is no way to implement safety against a user that leaves passwords in the open, allows access to his

or her system, or otherwise compromises the security of the application. It is also important to protect against any form of remote access by using a sound security policy.

The difficulty in ensuring secure end systems is increased by using a “softphone” system, such as the SIPcon1 application, which implements VoIP or similar services on an ordinary PC. The added security risks originate from the multipurpose abilities of a modern day computer, and the widespread existence of worms, viruses and other malicious software. Since other computer interactions might introduce these malicious programs without interference from the VoIP application no amount of VoIP application security can protect against these threats. Because of these external risks, a computer system requiring secure SIP communication needs to use good security measures to protect against those risks using antivirus software. It would be well to realise however that antivirus software is in no way a guarantee that a system is uninfected since even the best antivirus software can not protect against viruses so recently conceived that their existence is unknown.

### **4.2.2 Application Users**

One rather effective tool in hacking security systems today is what is known as “social engineering” which is basically the practice of conning people into giving you information sensitive to the security of a computer system. The fact that this method is so widely used is proof enough that no matter how secure an application is it is still no stronger than its weakest link, in this case that would be the user. Since no software can protect against this kind of attack it is important to remember to keep your users well informed on how to handle system sensitive information.

### **4.2.3 SIP Registrar and Registration**

The SIP registrar, as described earlier, is the entity that registers a SIP UA to its domain. The registrar usually requires the user agent to authenticate itself, otherwise it would be easy for an attacker to register, impersonating any legitimate user [37]. An attacker with the ability to register as any legitimate user can receive session requests intended for that user and may initiate requests to other users posing as the user of the hi-jacked account.

### **4.2.4 Location Service**

The location service, in whatever form provided, must remain inaccessible to unauthorized access. No amount of registration security would be sufficient if an attacker can modify registration data at the source. Since the location service most likely consists of a lookup on a database maintained by the registrar this simply means the database maintained by the registrar should remain inaccessible to unauthorized access.

### **4.2.5 Proxy Servers**

Because proxy servers are depended upon to deliver SIP messages these might become a popular target for attackers. Denial of service attacks by superfluous messaging can lock up proxy server resources and thereby prevent desirable traffic from reaching its destination [37].

A more serious security flaw would be for a legitimate proxy to become compromised in a way that allows an attacker direct access to the traffic routed through the proxy. This would allow the attacker to gain information on the contents of any message, even information

protected with hop-by-hop security, without alerting the users. If no end-to-end security is applied this would be the equivalent to being able to eavesdrop on all messages. Worse yet, if any cryptographic keys and parameters are piggybacked in the SDP message the entire media communication would become exposed.

Although a flawless security mechanism might ensure all messages come from an authenticated user at a specific domain it might be that the domain from which a message originates does not have a strict security policy on distributing licenses. This would mean that an attacker could register a legitimate user at one domain then proceed with flooding users of other domains with traffic spam, either for the purpose of performing DoS attacks or for the distribution of advertisement.

#### **4.2.6 Network Infrastructure**

Since any application using the internet to transfer information is vulnerable to attacks on the network infrastructure feasible steps to make the infrastructure less vulnerable should be taken. For example, proxy servers use a Domain Name Server (DNS) to route calls and are thus dependent on having a functioning DNS [37]. Though describing the particulars of these threats is out of scope of this document, thought should be given to how these threats can be countered.

#### **4.2.7 SIP Messages**

Sending unprotected SIP messages allows an attacker to eavesdrop on the communication learning everything about the upcoming session. SIP messages sent without any integrity protection could easily be modified by an attacker. It would also be possible to insert spoofed packets, modifying, ending an existing session or creating a new one.

The SIP protocol allows for the bye message of a session to be sent directly to the recipient as the location of the user, thanks to previously routed messages, is known. If hop-by-hop security is provided, it is highly recommended this traffic is sent the same way as the initial traffic since any bye-message sent unencrypted could just as easily have been created by an attacker. By inserting bye-messages to implementations susceptible to this form of session termination an attacker can easily perform a DoS attack, shutting down any session at any time by inserting a spoofed bye-message.

#### **4.2.8 Media Transfer**

When media stream has been successfully set up between two systems the stream of packets is vulnerable to a number of different attacks. The most notable security flaw of an unencrypted media stream is the lack of confidentiality. Anyone with access to the flow of information has the same access to the information as the intended recipient. With no integrity protection an attacker can choose to modify a packet, or send a faked packet with data chosen by the attacker, without the recipient being able to differentiate these packets from non-compromised traffic. For an experienced attacker it is trivial to insert faked packets into the network in a way that makes the receiver unable to distinguish real packets from faked ones when no check for authenticity is made.

Even in cases where confidentiality, authenticity and integrity is ensured it might be possible for an attacker to re-insert a previously sent packet in what is known as a replay attack so that the receiving application interprets it as a new, authentic packet.

The actual transfer of media is done with protocols and parameters decided in the SIP message exchange. The protocol used dictates what weaknesses the media stream might have. The SIPcon1 application, currently using RTP for the transfer of media, is susceptible to all the attacks described above.

## 5 Security Goals

To counter the threats recognised in the construction of the threat model several security mechanisms need to be employed. Threats targeting non-application specific resources, such as end systems containing a security configuration allowing legitimate access to a SIP account, can most often not be affected with implementation measures. Although the user SIP password might be left out of a configuration, introducing user supplied passwords, this would still mean the user could leave the application running on an unguarded system, leave a note with the password lying around or give it to an attacker exercising social engineering. These types of threats can ordinarily only be countered by using good security policies and educating application users.

Some non-application specific threats, specifically DoS attacks targeting infrastructure resources, such as proxy servers, can be mitigated by a good network infrastructure and using strict policies on accepting SIP requests coming from certain sources. For example, a proxy knowing that some traffic originates from callers associated with a domain using lax registration policies might disallow that traffic. Although good security practices can serve to reduce the effects of DoS attacks [7] it is difficult, or impossible rather, to be completely safe from such attacks.

There are a few common goals attainable by securing any transfer of information requiring some form of protection by using encryption:

- Message authenticity
- Message confidentiality
- Message integrity

In addition to these goals it is important to prevent any legitimate message reinserted into the network from being accepted as a new message. This feature is called replay protection.

To provide the necessary features to both the SIP communication and media streams each of these features needs to be implemented by some form of encryption protocol. For each of these encryption protocols there needs to be some means of negotiating keys for that encryption. Thus the problem consists of providing:

- Authentication of users so exchange of encryption information (key agreement) can be guaranteed to take place between intended participants of the session only.
- Key agreement for every security protocol which requires it.
- Security protocols to provide required features using the acquired key agreement data.

While this chapter only describes common security features and the motivation for their importance later chapters will provide information on protocols which will make the realization of these features possible.

### 5.1 Registration

The first communication that needs to be secured in a SIP based application is the registration of the client to its registrar. This is the foundation on which all other security measures depend. If the registration process is insecure there can never be any guarantee that a session participant is who he or she claims to be. An attacker registering his own device/system as the contact of a legitimate URI is performing what is known as registration hijacking. In this case all other security measures would be compromised. If registration hijacking can be performed on one URI it can be performed on two as well. In a session where two users have access to a registrar without secure registration it might be possible for an attacker to re-register both legitimate user systems as the correct contacts for two other URIs and register his or her own system as the appropriate contact for those users. This way the attacker could route all traffic between the two users through his system allowing total, man-in-the-middle access to the SIP information.

### **5.1.1 SIP Registration Authentication and Integrity**

When connecting to a registrar the user needs to be able to ascertain the true identity of the reached entity. In other words that the entity reached is indeed the intended registrar. Likewise the registrar needs to be certain the register request is being sent by the correct user.

If a registrar does not require authentication from a user client, or if the authentication required is too weak, the entity is subject to a number of attacks. First of all an attacker could register as any legitimate user, de-registering existing contacts for that URI, registering their own devices as a contact address. This way they could hijack all SIP traffic intended to go from or to the legitimate user.

Having a weak or non-existing authentication of the registrar also presents additional security issues as it would allow an attacker with access to the UAC-to-registrar traffic (a man-in-the-middle access) to impersonate the registrar. An attacker faking a registrar could respond with a 301 (Moved Permanently) message to make all future requests go to an address chosen by the attacker. A client allowing authentication negotiation could find itself facing a downgrading attack where the attacker, posing as the registrar, tries to offer an authentication mechanism that either exposes secret credentials (password) or is easily broken.

The current implementation of SIPcon1 only uses the predefined HTTP digest authentication [12, 37], where a user password is used with MD5, to authenticate the client to the server. A unique nonce and a nonce counter are used to prevent replay attacks where an attacker could otherwise have re-sent previous authentication data in a new request. Digest authentication allows authentication without exposing secret credentials in the SIP communication and because of the way the authentication process is performed allows the registrar to authenticate a UA using a hashed value, without having to store the password explicitly. This way an attacker gaining access to the registrar database would not learn the passwords with which to create spoofed registrations. Note that the current implementation of digest authentication in SIPcon1 only authenticates the client to the server, not the other way around.

It is desirable to protect messages in such a way that a message can not be altered without detection, preserving message integrity. A registration procedure that does not guarantee message integrity runs the risk of having message values altered. While digest authentication has no explicit integrity protection some of the information in the register request is implicitly protected due to the simple fact that it is used as input in computing the hash. For message

tampering to be detected on registration messages an implementation needs to provide some form of additional integrity protection.

### **5.1.2 SIP Registration Confidentiality**

Allowing registration in plaintext allows an attacker to gain all information about the requests and responses sent. Although, in the case where http digest authentication is being used during registration, this might seem like a minor problem it still allows an attacker to monitor SIP traffic. A packet recorded in this manner could be used to launch a replay attack later on, should the registration procedure not include a mechanism for preventing such attacks. Fortunately, as previously mentioned, digest authentication provides mechanisms preventing replay attacks. For SIP registration to ensure confidentiality the current registration process would need to be replaced or carried out over an encrypted connection.

## **5.2 SIP Communication**

There are several aspects to be considered when trying to provide security mechanisms for session establishment and configuration. There needs to be mechanisms in place to ensure a secure session can be successfully set up without an attacker being able to eavesdrop on the communication, modify messages or create spoofed messages. As with the SIP registration process the three main goals of adding security to the communication is to ensure message authenticity, integrity and confidentiality. The main difference between SIP registration and other traffic is that SIP registration only needs to deal with local, known users and their certificates while other requests often have to travel through proxies not of the same domain.

### **5.2.1 Session Message Authenticity and Integrity**

It is important to ensure all SIP traffic originates at the expected source and that it has not been altered. Although secure SIP registration ensures that calls received by an inbound proxy, intended for the registered recipient, reaches the right destination, registration alone provides no insurance of the traffic origin. In other words, an outbound proxy that requires no authentication of origin on invite messages sent from its domain allows attackers to create spoofed requests by sending messages to that proxy posing as a legitimate user. An attacker could feed such a proxy falsified requests and thereby use that proxy to launch attacks that would seem to originate from one of the domain URIs. An end user will also need to authenticate received messages as an attacker could spoof a message so that it appears to have been delivered through the proxy. An inbound proxy not authenticating incoming messages would present the same threat to the users of that domain because any incoming messages, thought to originate from another legitimate user or proxy, could in fact be spoofed request sent by an attacker posing as another domains proxy.

To make sure this threat is neutralized there needs to be a mechanism that ensures message authenticity either using a mechanism similar to the one used in the registration process [12] returning a challenge for every message or by sending messages over a previously established secure connection or association [37]. There is also the possibility of adding end-to-end authenticity to the message, requiring both participants to authenticate themselves to each other.

Authentication between proxies needs to be considered a slightly different problem than user to proxy authentication. A process similar to that of the currently implemented user-proxy authentication is not viable between proxies because that solution depends on the proxy being a central authority (maintaining passwords) and does not scale well. Instead, proxy servers need to make use of some type of certificate authority to authenticate each other. Another method would be for each proxy pair to sign each others certificates, but while negating the need for a trusted external CA this method does not scale well.

By computing a hash, based on some secret credentials and session message content, the content used in the hash becomes integrity protected as any modification would alter the result of hashing at the receiving end. This means that if the receiver of a message positively authenticates the sender, provides mechanisms that ensure no replayed messages are considered valid, and computes a hash over the message using some secret credentials shared with the sender, any SIP message received where an included hash result matches the computed hash of the message will be an authentic, unaltered message.

If any entity along the SIP route should need to change information in the SDP body of the message, possibly to disallow certain types of media streams or for any other reason, end-to-end authentication of the SIP message would prevent this as any tampering would lead to the receiver interpreting it as an invalid message. The only way in which tampering would be possible would be if an entity part of a hop-by-hop security, where no end-to-end authenticity is applied, changed the contents of the message. Although this would promote interoperability with certain types of firewalls, one of the main concerns with using exclusively hop-by-hop security is precisely that security is only maintained if it can be ensured no intermediate proxies are compromised or malicious. The primary motivation for using end-to-end authentication is that any such tampering would be detected at the end destination.

Authenticating call origin is pivotal, in the same way as secure registration, because anything else would make not only the compromised domain vulnerable, but indeed the entire SIP infrastructure. Domains that are unable to guarantee message origin, or unable to ensure secure registration, are likely to have messages routed through them ignored by other domains because vulnerabilities in one trusted domain compromises one's own domain. Such flaws would thus not only compromise the local SIP functionality but would also prevent communication with other domains.

### **5.2.2 Session Message Confidentiality**

Confidentiality of the SIP messages is needed to prevent an attacker from learning session specific information. An attacker will be able to monitor participants of SIP sessions, the time during which they communicate with each other, what types of media they communicate with, what ports and transport protocols are used for the transmission and any other session specific information. Because the SDP message body can be used to convey cryptographic information regarding the media transfer when the media encryption protocol itself does not provide these services (as is the case with SRTP), the protection of the SIP content under these circumstances becomes crucial to ensure the security of the media streams is maintained. Tunnelling (piggybacking) this information in the SIP message body is a convenient way to perform key agreement, but these schemes also make the media security totally reliant upon SIP message security to maintain their usefulness. This is an example of yet another process where another weak link in the cryptographic chain could compromise the entire cryptographic process. As in the process of ensuring message authenticity, solely

relying on hop-by-hop security could constitute a security risk depending on how safe involved proxies (or other SIP entities) are rated. This is why it might be prudent to apply end-to-end encryption in some security models. Some of the SIP header fields are unsuitable to encrypt end-to-end as their content is needed by intermediate proxies for routing purposes, which is why end-to-end encryption of SIP messages should either encrypt the SDP body or the body including header fields not required for routing. While there are certain entities today which require access to the SDP information when routing a session message they are generally rare and the benefits of adding end-to-end encryption in some cases overshadow this disadvantage.

### 5.3 Media Transfer

Securing the actual transfer of media is a problem in many ways unrelated to securing the SIP messages. The SIP message security and media transfer security share one important common link in the need for negotiation. Since the purpose of the SIPcon1 application security mechanisms is to be made compatible with other SIP/SDP based applications, and since there are no “one size fits all” standards to cryptography, negotiation of keys, transport mechanisms and cryptographic suites would allow for maximum compatibility. The SDP body of a SIP message is the perfect medium for such negotiation and it is therefore expected solutions catering to the need for interoperability will probably make use of the SDP body in negotiating parameters for the media encryption. An unavoidable problem with this approach is that for this negotiation to work both parties must have implemented the protocol required for this negotiation. Since information on encryption parameters leaked to an attacker would render the encryption useless any scheme using the SDP information for that purpose is dependant on the SIP security. Also, since the security protocol using these parameters will need to be negotiated tampering with this part of the SDP information could mean the security mechanism becomes compromised as an attacker could launch a downgrade attack where a less secure, or totally insecure, transport protocol offered as a secondary alternative is left as the only offered protocol.

Ordinarily, securing the media for a VoIP application would set high QoS requirements for the encryption since user perception of audio and video media is sensitive to QoS issues. The focus herein lies on the transfer of text media, not at all as sensitive to these issues as audio and video, but to allow for future extensions to the encryption considerations to QoS issues will still be prioritized.

#### 5.3.1 Media Authenticity and Integrity

A protocol relying on UDP for the delivery of packets takes a packet arriving at the expected port and processes it. An attacker could insert faked packets into the network appearing to belong to the media session to force a receiver application to process data created by the attacker. Using this method an attacker can easily degrade the QoS to such an extent that communication becomes impossible. A more subtle attack on a media stream without authentication and integrity would be to intercept and modify media packets in such a way that the meaning of a voiced sentence, transmitted video or text changes completely.

Providing the session initiation was secure participants of the session have had an opportunity to exchange cryptographic keys and parameters known only by them. Using these credentials to create a hash over the media in need of protection and including the result in the packet

data would not only prove to a recipient the message originated from the expected sender, but it would also prove any packet information needed to create the hash has been left untouched.

An attacker faced with a confidential, authentication and integrity protected message could still cause damage to some applications. If no mechanism is implemented to prevent the re-use of old packets an attacker could record packets and resend them as though they were legitimate packets. This attack is called a replay attack and any transport protocol concerned about security should provide mechanisms to prevent this attack.

### **5.3.2 Media Confidentiality**

Once the media transport protocol and required parameters have been negotiated the actual media stream needs to be made confidential. By encrypting the media with secret cryptographic parameters shared with the recipient any eavesdropping would produce nothing but seemingly random data.

As the quality of real-time transferred media at the receiving end is degraded by the effects of packet latency, jitter and packet loss steps must be taken to reduce these QoS issues as much as possible. Packet loss is possible since the time constraints on real-time media make it unsuitable for reliable transport protocols to be used, but no encryption mechanisms add to the problem of packet loss except perhaps to increase the chance a packet arrives to late for proper processing. Latency and jitter of packets are two of the QoS aspects affected by encryption. The computational time needed to encrypt and decrypt a packet of media data is added to packet latency, and because the time needed for this process may vary depending on the current performance of the system the jitter effect is also increased.

The importance of media stream confidentiality and QoS sensitivity suggests encryption needs to be both fast and difficult to break. The possibility of packet loss in UDP based network delivery, for example RTP, requires the encryption to be stateless since any encryption based on previously transmitted data would become out of synch as soon as a packet was lost.

## **III Security Solutions**

## 6 Encryption Protocols

One encryption protocol can be used in different contexts for slightly different purposes. This chapter describes some of the more common encryption protocols for encryption at different layers. The utilization of these protocols to solve certain encryption problems will be discussed in chapters, which handle encryption in a particular context.

### 6.1 IPsec: Network Layer Encryption

The IP Security Protocol suite, known as IPsec [24], consist of a number of documents, interrelationships explained in [39], that provide security at the IP level (in the network layer). The preferred form of VPN tunnelling across the internet [27], IPsec defines two basic protocols which provide connectionless integrity, source authentication, confidentiality and replay protection: Encapsulating Security Payload (ESP) [23] and Authentication Header [22] (AH). Given that IPsec used in tunnelling mode can provide these services to an entire IP packet, including the header, it is able to provide better security than encryption at other levels. IPsec is most commonly used where domains or hosts have an existing trust relationship with each other [37]. It can be used regardless of the transport protocol being used and provides authentication, integrity and confidentiality for transmitted data.

The SIP protocol RFC does not specify a framework for the use of IPsec and no key management is suggested. The most common use of IPsec in this scenario is to use it in collaboration with the IKE protocol [14]. Based on the ISAKMP [28] framework IKE uses the Oakley Key Determination Protocol [31] and SKEME (Secure Key Exchange Mechanism for the Internet) [25] to provide automated cryptographic key exchange and management mechanisms.

Two protocols ESP (Encapsulating Security Payload) and AH (Authentication Header) can be used in either transport mode or tunnel mode [27]. Transport mode is used to encrypt the payload data leaving the IP header and the new IPsec header in the clear. While being the faster of the two methods this mode of delivery leaves an attacker the opportunity for rudimentary traffic analysis. Tunnel mode not only encrypts the payload, but the entire IP datagram and places the encrypted data in a new IP Packet. This way only the entity currently sending the Packet and the next endpoint in the tunnel are made known to any eavesdroppers, making the true origin and final destination unknown to an attacker. The cost for this added security is in the added latency of encrypting and decrypting the original IP header and the added size of the transmission due to the extra IP header.

The SIP specification states that “IPsec is usually implemented at operating system level in a host, or on a security gateway that provides confidentiality and integrity for all traffic it receives from a particular interface” [37]. Because operating systems are trusted to implement IPsec and because existing trust relationships should be present IPsec is not suited for deployment in all network environments. IPsec also has known issues with NAT traversal that vastly increases the difficulty of deploying a working IPsec security mechanism. Thus an application relying on IPsec to secure its communication is also more suitable for deployment

on a local, known network basis. The NAT traversal issues are derived from the need for a NAT to translate between inner and outer addresses on a packet with encrypted headers.

Although suitable for file transfer scenarios the overhead to packet size and computational time of IPsec is rather hefty in terms of real-time transmission. Applications that require minimal delay and jitter will in many cases find the time required for IPsec encryption unacceptable.

## 6.2 TLS: Transport Layer Encryption

Transport Layer Security (TLS) [9] is a security protocol which provides encryption at the transport layer. The protocol specifically requires a connection oriented, reliable delivery transmission protocol which means it will not work with protocols using UDP transmission. TLS provides integrity protection, authenticity and confidentiality of sent data without needing additional key management.

TLS is comprised of two layers; the TLS Record Protocol and the TLS Handshake Protocol. The Handshake Protocol is used to authenticate the participants and negotiate security parameters while the Record Protocol provides confidentiality and integrity to the actual data transfer [9].

Although commonly used for http security (using the https URI) TLS is application independent, meaning it can be used with a variety of applications using connection oriented, reliable data transmission. How applications or protocols are supposed to use TLS handshaking, authentication and so forth is left to the protocols which run on top of TLS [9].

Because TLS is application independent, self-sufficient and widely used it is a good alternative to consider for any situation where TCP based traffic needs encryption. By having a certificate for the entities involved in the communication TLS provides its own key management and authentication using the handshake protocol before proceeding with the encryption (using the record protocol).

A draft is being worked on to create version 1.1 of the TLS protocol [10]. The specifics of this document have not been considered in the creation of this thesis, but the completion of the protocol could very well introduce some interesting possibilities in the future.

## 6.3 SRTP: Application Layer Encryption

The Secure Real-time Transport Protocol (SRTP) [6] can offer confidentiality, authenticity, integrity and replay protection for RTP and RTCP packets, providing all the important elements to secure a media stream. The RTP packets are used to carry the session contents while the control packets, RTCP, are used for session statistics and control. A secure session key derivation function is used to produce pseudo-random session keys using only a master key and an optional (highly recommended) master salt.

The key derivation function in SRTP enables session keys to be created using a master key and, to protect against pre-computation attacks, a master salt. This function is used to create a session encryption key, session authentication key and session salt to use when processing

packets. In fact, both the SRTCP and SRTP stream can be provided with session keys using only the master key and salt. This function also enables the definition of the optional key derivation rate in the SRTP protocol which specifies how often new keys are to be generated. This is useful because an application sending data for a long period of time might wish to use several session keys so that one leaked or cryptographically broken key will only compromise part of the packet stream. This is then easily done by defining a key derivation rate higher than zero.

The construction of an SRTP implementation has been constructed in this project since it proved to be a very effective encryption method for the encryption of real-time media. Because of this a slightly more through examination of the protocol is warranted.

SRTP uses the fast and strong Advanced Encryption Standard (AES) [30] (based on the Rijndael cipher) in counter mode as the default algorithm for encryption and proposes AES in f8-mode as an option for UMTS (Universal Mobile Telecommunications System) applications. The counter mode of the AES cipher works by using an initialization vector, computed with the source identifier, packet index and the salt. This IV is then used to create a keystream for the first “blocksize” number of bits of the packet. Any further keystream needed for that packet is created by applying AES to the IV plus a counter incremented for each block. The keystream produced is XORed with the plaintext datastream to produce the encrypted stream and vice versa for decryption.

The authentication mechanism provided supports both message authentication and message integrity using the default HMAC-SHA-1 (HMAC [26], SHA-1 [29] transform. The authentication mechanism is also what paves the way for the use of the replay protection mechanism and is required for replay protection to work.

The SRTP protocol leaves a small footprint by requiring a small code size and little memory and as the packet expansion is small bandwidth is preserved. By conserving the headers of the RTP packets any header compression used with RTP can also be used with SRTP.

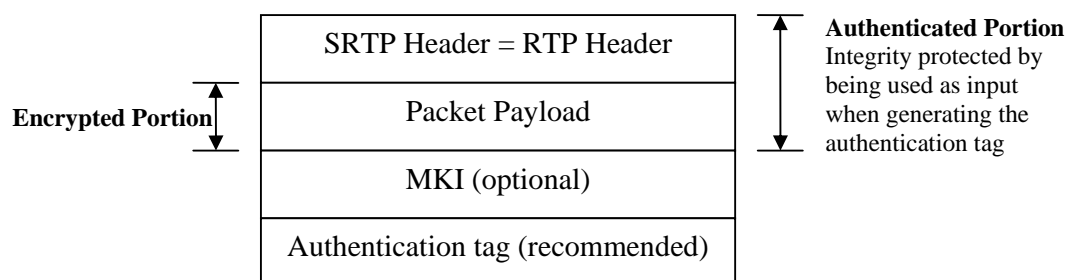


Figure 6.1: SRTP packet composition

The SRTP header, on the top in Figure 6.1, is identical to that of an RTP header. The difference between the SRTP and RTP packet is that the payload of the original packet is encrypted and that two optional fields may be added.

The intended purpose for the optional MKI (Master Key Identifier) field is to identify which master key has been used to derive the session keys to encrypt the packet. This field is defined, signalled and used by key management. The purpose of changing a master key might be because the master key can no longer be trusted. There might be a suspicion that the master key in some way has been compromised in which a re-key would have to be made. There is also a limit to how many packets can be encrypted with the same master key. As a new master key is negotiated in every new SIP request the first use of the MKI should not be needed. The packet limitation on the master key requires a time period of several months of intense communication to be transgressed and should pose no problem to an application used for short conversations.

The Authentication tag for an RTP packet is used to provide authentication, integrity and the basis for replay protection. Even though the tag is optional its inclusion is highly recommended by the SRTP specification. All applications wishing secure communication using SRTP should make use of the authentication feature, otherwise only confidentiality can be guaranteed. The tag is computed using the session authentication key and, with the default algorithms, the HMAC-SHA-1 transform over the original RTP header and its encrypted payload. This way the receiver can compute the tag of a received packet and compare it to the appended tag. If the tags are a match the packet has been sent from a party with the correct cryptographic context, without any of the integrity protected content being manipulated.

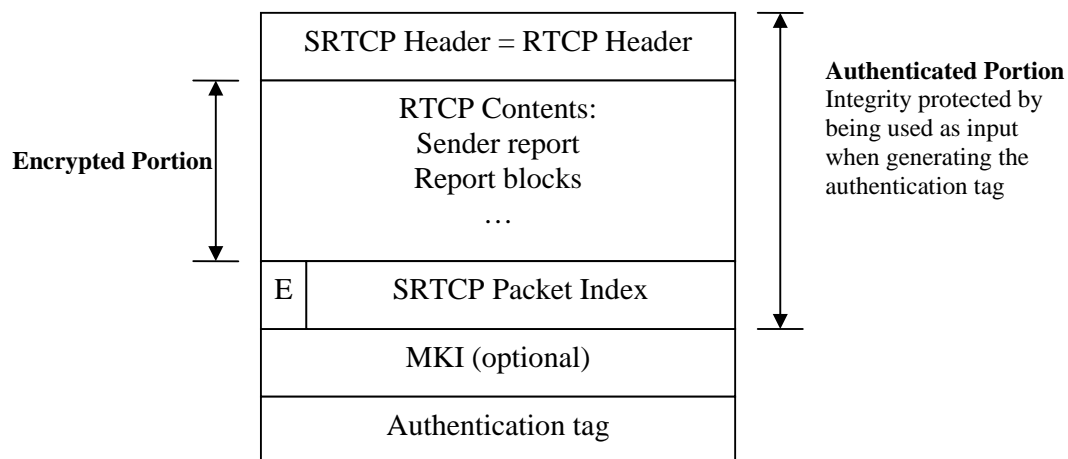


Figure 6.2: SRTCP Packet composition

The SRTCP packet, shown in Figure 6.2, introduces some new fields much like the SRTP packet does. The difference here is that that an explicit packet index and encryption bit (the E in Figure 6.2) is added, and that the authentication tag is mandatory.

<b>IPsec</b>	<b>TLS</b>	<b>SRTP</b>
Can be used with TCP or UDP	Can be used with TCP	Used with RTP and hence requires UDP
Able to protect an entire IP packet	Layered on top of a reliable transport protocol	Able to encrypt RTP/RTCP payloads and protect the integrity of RTP/RTCP headers.
Some NAT traversal problems	Adds no NAT traversal issues	Adds no NAT traversal issues
Large packet overhead produces large packets	Mature protocol with good availability due to SIP recommendation	Small increase in packet size
Implemented at operating system level (or in a security gateway)	Easily supplemented with end to end security mechanisms due to easy interaction between the application and TLS	Fast encryption
Relatively new protocol		Requires external Key Management
Requires key management (IKE)		Able to generate many sets of session keys by use of a master key and master salt

Figure 6.3: Overview of some basic Security Protocol traits

## 7 Media Encryption

The main purpose of adding security mechanisms for the transfer of RFC2793 [16] payloads is to be able to make all transferred media, and nothing other than authentic media, accessible only to the intended recipient. All other security measures described are needed because a secure media transmission cannot be set up without them or because they provide some additional security features, like preventing traffic analysis. The successful application of security features to the transfer of media requires that participants can rely on the fact that:

- The other participant has been properly authenticated
- A secure transport protocol supported by both parties has been agreed upon
- A cryptographic suite supported by both participants has been agreed upon
- Keys and other required cryptographic parameters have been securely agreed upon
- All requirements for a non-secure media transmission have been met

If any of these requirements are not met by preceding security measures they have to be ensured by the media transfer security itself.

The security of the media transmission may be the main purpose of all preceding security mechanisms, but this does not mean the importance of the security mechanism encrypting the media is any greater than the other security mechanisms. Any security mechanism required for the media encryption to be useful that is compromised, such as key exchange, might indirectly compromise the media security.

In addition to the requirements for creating a secure transmission of the media stream this part of the security mechanism in particular has high performance requirements. Since QoS issues arise easily in streaming media any method for encrypting the media must provide quick as well as strong encryption. Because of the time constraints associated with encryption of a live media stream, symmetric-key cryptography is more suitable than public-key cryptography.

The following sections deal with some of the most common ways of encryption on different levels of the stack, describing advantages and disadvantages of these approaches in the context of real-time media transmission.

### 7.1 IPsec: Network Layer Encryption

The ability of IPsec [24] to mask the address of the intended recipient is useful to prevent traffic analysis where an attacker might gain information such as communication participants and call length. Another reason to consider IPsec for media encryption is that the protocol in some cases might be used to secure the SIP messages and hence implementation of a new secure transport protocol would not be needed. This is especially true since IPsec might already be used in a VPN context. Since IPsec will be supported by the protocol stack in IPv6 it could be worth to consider using it to ensure minimal changes to an application need to be made in the future. The protocol does however suffer from a number of disadvantages as well.

One of the disadvantages comes from IPsec currently being implemented by the operating system. This forces the user agent to interact with the operating system when setting up a new security association and requires that the receiving systems operating system also supports IPsec. This not only makes an implementation complex and operating system dependent, but it also introduces a dependency on the operating system to keep the cryptographic information secure.

Because use of IPsec for encryption of the media stream would require two end systems to share a Security Association (SA), IPsec alone would be unsuitable for purpose of communication between two end systems with different setups. This problem is usually handled by implementing the Internet Key Exchange (IKE) [14] protocol for the purposes of handling management mechanisms, which means another protocol needs to be supported and used by both participants. This could lower the chances of interoperability, but using IKE in this way with IPsec is fairly standard and any application using IPsec should be able to use IKE to determine these parameters.

Setting up everything needed for IPsec communication introduces latency to the call setup time. According to NIST-sponsored testing [27] IPsec can be incorporated into a SIP network with about a three-second delay added to the call setup time which could be acceptable to some applications. However, considering the importance of QoS in an application with VoIP characteristics any added latency of that magnitude is best avoided.

When tunnelling mode is used IPsec adds a substantial amount of overhead to the packet size. In addition to the actual packet an AH, ESP and an additional IP header is added. This overhead can have the effect of increased latency and jitter on the packets in addition to requiring more bandwidth. Some degradation of the QoS for an application using only text communication may not be too detrimental to the application's functionality, but should audio and video transmission be subject to such QoS obstacles the effect would most certainly be devastating.

Using IPsec for end-to-end encryption is known to cause problems with NAT (Network Address Translation) and firewall traversal. Firewalls might block packets where encryption of the ports would indicate a port where traffic is prohibited was trying to be reached. Since UDP headers need to be changed at the NAT, and the UDP headers themselves are encrypted in IPsec, Nat traversal becomes complicated. There are solutions to this problem, most notably UDP encapsulation of IPsec [19]. By using IKE negotiation and adding a new IP and UDP header using port 500 (IKE port used so that no new holes need to be punched in the firewall) and setting the SPI field of the encapsulated packet to zero (to differentiate it from IKE communication) encrypted media packets should be able to travel through NATs in both directions. Implementing this method means adding complexity to the security mechanism and, in addition to increasing packet size, adds more overhead to the process, further complicating QoS issues.

As the drawbacks of using IPsec when securing the media stream have shown, the protocol is not really suitable for the purpose of real-time media encryption. Also, the sheer number of protocols involved and the loose definition on how or whether to apply some of these protocols makes implementation, particularly considering compatibility issues, and analysis of the security configuration difficult. However, there are indications that, since IPsec is very useful in some situations, future technology and standards could pave the way for the use of IPsec even in real-time media situations. Even if, right now, IPsec may not be a viable

alternative in an application such as SIPcon1 consideration should be taken to the fact that future developments might change this fact.

## 7.2 TLS: Transport Layer Encryption

As the SIPS (Secure SIP) definition in the SIP specification implies the use of TLS [37] there is a great probability that an application securing SIP messages makes use of TLS. This might make TLS seem like a good alternative for encryption of the media stream because no other security protocol would need to be implemented. This would be a false assumption since TLS uses the reliable transport protocols, such as TCP [9]. While this might not be a great problem where SIP messaging is concerned it is devastating in the context of real-time media encryption. The nature of streaming media simply does not allow the use of a reliable delivery protocol such as TCP for transportation due to time constraints.

## 7.3 SRTP: Application Layer Security

When implementing application level security for transfer of real-time media, SRTP [6] is without doubt the most appropriate protocol to use. Not only was it conceived specifically for the purpose of distributing real-time data, but it is also defined as a profile for RTP making it easier to implement with an RTP implementation already present.

Since no key management is defined in SRTP, external key management mechanisms are used to exchange keys and cipher-suite information and parameters. This makes SRTP used within a SIP context reliant on external key management, for example using predetermined keys and parameters, a predetermined key management protocol or SDP negotiation. Because of this dependency, and because SRTP is such a useful protocol for secure media transmission purposes, the SDP extension protocols [1, 3] described in Chapter 9 are geared towards providing this information to SRTP. This is not to say they cannot be extended for use with other protocols, but the intended use upon their creation was with SRTP.

By requiring external key management using SRTP creates the requirement for an encrypted exchange of key data which cannot be satisfied by SRTP itself. However, since SDP tunnelling of this information is possible the encryption of SDP data or SIP messages, which should be present, would also serve to protect tunnelled key agreement information. The benefit of doing this, in addition to piggybacking on the SIP security, is that using the SDP information makes configuration of or changes to the security mechanisms very easy to implement.

## 8 Key Agreement

All types of encrypted communication require both participants to agree upon how to perform encryption and decryption. Specifically, participants need to know what crypto-suite, keys and crypto-suite specific parameters are being used. To provide these services to protocols that do not themselves cover this area (such as SRTP or IPsec) key management or key agreement protocols are needed.

### 8.1 Standard Key Agreement protocol

When it comes to exchange of the actual keying information there are a few well known and common mechanisms used within key management protocols. These mechanisms are needed because of the paradox that key information must not be sent in the clear but the information itself is needed for encryption. Exactly how these mechanisms are applied is up to the key management protocol that uses them, but the basics of the methods are described below.

#### 8.1.1 Pre-Shared Key Agreement

The most basic of key agreement protocol, pre-shared key requires that both participants share a secret key. Both participants use this key to derive an encryption key (possibly also an authentication key) and the initiator creates and sends a randomly generated session key encrypted with the generated encryption key. The recipient receives the message and decrypts it using the symmetric encryption key and thus gets access to the session key included in the message.

While this mechanism might be fast and simple it suffers from the requirement of maintaining a shared secret key with an intended participant. This scheme does not scale well because each user would need to maintain a key for each possible recipient, in addition to requiring some means of obtaining these keys. A pre-shared key scheme is therefore only suitable in a scenario where few users need to communicate and key distribution is easily handled.

Because a pre-shared key can be used for a long period of time, and because refreshing this key can be arduous, many key exchanges can be performed using the same key. What this means is that if the secret key is disclosed to an unauthorized party all future transactions using that key are compromised until the key is replaced.

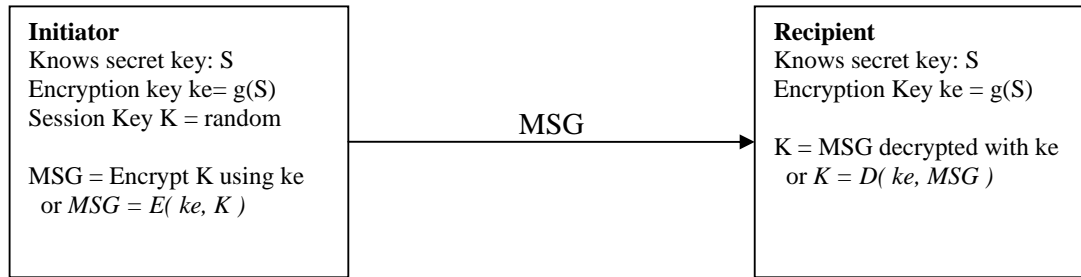


Figure 8.1: Pre-Shared Key agreement protocol

### 8.1.2 Public-key Agreement

Public-key encryption uses a key-pair, one public key and one private key, to encrypt messages. A message encrypted with the private key can only be decrypted with the public key and vice versa.

The public-key encryption can be used to exchange the needed information with what is known as a digital envelope. By using the public key of the responder the initiator can encrypt the generated secret key so that only someone with the recipient's private key (hopefully only the intended recipient) can decrypt the content. By signing the message using his or her private key the initiator provides authentication of the message, since the responder can use the initiator's public key to make sure the message was signed by the sender.

Public-key encryption adds the opportunity of distributing keys using a PKI at the expense of an increase in computational cost. The public keys needed for the exchange can be disclosed without compromising future key exchanges. If a private key is disclosed however, all future transactions, and indeed any recorded old transactions, using that key pair become compromised.

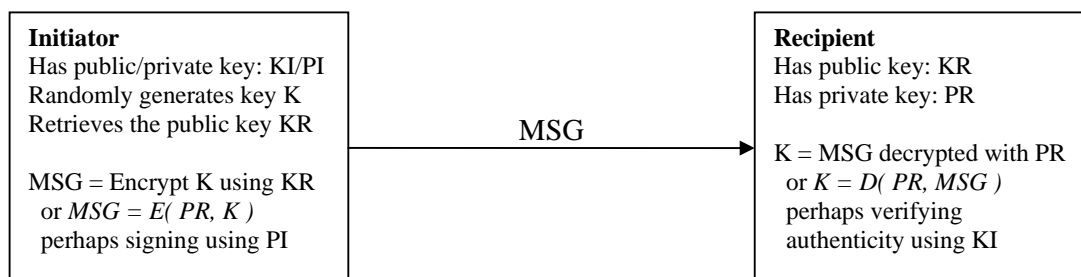


Figure 8.2: Public-Key agreement mechanism

### 8.1.3 Diffie-Hellman Key Agreement

The Diffie-Hellman key exchange [36] is the only type of key agreement protocol which provides perfect forward secrecy, meaning a disclosed key does not compromise future exchanges. This is possible because the Diffie-Hellman key exchange does not rely on an existing key to provide secrecy. Rather, this method relies on the complexity of a

mathematical problem called the discrete logarithm problem to generate a key for each transaction and therefore no predetermined key is needed. A drawback of this method is that it is usable only in a peer-to-peer situation, but in the context of a peer-to-peer SIPcon1 session this limitation is unimportant.

First the participants must decide upon Diffie-Hellman parameters, which consist of a prime number  $p$  (larger than 2) and an integer smaller than  $p$  denoted  $g$ , usually called a generator. These values are signalled by the initiator of the exchange.

Both participants generates a secret private number  $x$ , which must be smaller than  $p-1$ . The initiator uses his generated number  $x_i$  to calculate a public key  $y_i$  (Equation 8.1) which he then sends to the responder.

$$y_i = g^{x_i} \text{ mod } p \quad (\text{Equation 8.1})$$

The responder calculates a public key in the same way, using his own private number  $x_r$  (Equation 8.2) and responds with this key.

$$y_r = g^{x_r} \text{ mod } p \quad (\text{Equation 8.2})$$

What we want is a secret key shared by the key exchange participants but indiscernible by an eavesdropping attacker. A common secret key  $z$  involving both private keys can be calculated using Equation 8.3.

$$z = g^{(x_i \cdot x_r)} \text{ mod } p \quad (\text{Equation 8.3})$$

An attacker will only have access to the public keys and each participant will lack one of the secret keys needed for Equation 8.3. The difference is that legitimate participants can use their secret key along with the public key, sent by the other participant, to calculate the common secret key which can be used to generate keying material. Equation 8.4 shows the equation relationships which makes this possible.

$$z = g^{(x_i \cdot x_r)} \text{ mod } p = y_i^{x_r} \text{ mod } p = y_r^{x_i} \text{ mod } p \quad (\text{Equation 8.4})$$

Again, an eavesdropper to this conversation would get access to  $y_i$  and  $y_r$  (the public keys), but since it is considered computationally infeasible to derive an  $x$  value given the corresponding  $y$  value an attacker would be unable to obtain the secret  $z$  value. The entire process is illustrated in Figure 8.3.

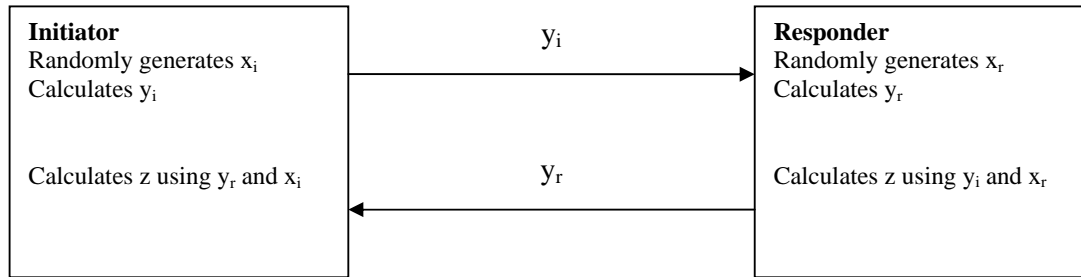


Figure 8.3: Diffie-Hellman key agreement

The Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack. In this scenario an attacker would intercept the initiator's message and respond as though he or she were the intended recipient. The attacker could then initiate another Diffie-Hellman exchange to the original recipient as though he were the original initiator. The result will be that the attacker establishes two different keys with the initiator and responder after which all subsequent messages between the two original participants can be decrypted, possibly modified, and re-encrypted by the attacker. Even though man-in-the-middle access can be difficult to obtain in practice this must be considered a serious security flaw.

To counteract the man-in-the-middle vulnerability the usual solution is to provide authentication over the exchange. By using a private key (of a public-key pair) to cover the public value of  $y$ , the sender of that message can prove to the recipient the value has not been altered. This defeats the man-in-the-middle attack but also introduces the need to handle and validate certificates using a PKI.

## 8.2 Key Management Protocols

Key management protocols are protocols which incorporate key agreement mechanisms to handle cryptographic parameter agreement. By using the format of the key management protocol the necessary parameters needed for two participants to be able to set up a secure communication can be agreed upon.

### 8.2.1 ISAKMP

The Internet Security Association and Key Management Protocol [28], ISAKMP, is not really a key management protocol, but rather a framework for how to implement key management protocols. The protocol abstract states that “[*The protocol*]... defines the procedures for authenticating a communicating peer, creation and management of Security Associations, key generation techniques and threat mitigation”[X-28].

### 8.2.2 IKE

The Internet Key Exchange, or IKE, is a hybrid protocol based on ISAKMP, the Oakley Key Determination Protocol [31] and the Secure Key Exchange Mechanism for the Internet [25] (SKEME). IKE was developed specifically to handle the key agreement needed for the IPsec DOI (Domain of Interpretation).

The IKE exchange is comprised of two distinct phases. The first phase establishes a secure, authenticated channel with which to communicate (using the ISAKMP SA) while the second phase negotiates a SA (Security Association) needed by another protocol using the channel established in phase 1. The first phase can be executed in either Main Mode or Aggressive Mode while the second phase is executed in what is called Quick Mode.

Main Mode is an instantiation of the ISAKMP Identity Protect Exchange and requires six messages, in three roundtrips, to be sent. The first two messages negotiate cryptographic information, i.e. crypto-suite and parameters, using an ISAKMP SA payload. The next two messages exchange the Diffie-Hellman values with a KE and an ancillary payload. The last roundtrip provides authentication of the Diffie-Hellman exchange by the inclusion of identification information under the protection of the common shared secret. By separating the key exchange from the authentication information the identities of the participants are protected by the common shared secret at the expense of additional roundtrips.

Aggressive Mode is an instantiation of the ISAKMP Aggressive Exchange, requiring 3 messages. This mode transmits the SA, key exchange and authentication payloads together in one message, thus reducing the number of messages that need to be sent. While cutting the number of required messages in half, this mode does not protect the identity information. The first pair of messages exchanges the combined information package while the last message, sent by the initiator, serves to authenticate the initiator to the responder.

Four authentication methods are allowed with Main Mode and Aggressive Mode:

- Digital signature, where the exchange is authenticated by signing a mutually obtainable hash.
- Two forms of public-key encryption authentication which provides plausible deniability of communication at the cost of added computational expenses.
- Pre-shared key authentication, where participants need to share a common secret key.

Quick Mode used for phase two of an IKE exchange uses the already established ISAKMP SA to protect the information exchange, encrypting all payloads except the ISAKMP header. Quick Mode provides SA negotiation and nonce exchange for replay protection with an exchange comprised of three messages.

Because IKE uses a Diffie-Hellman exchange to negotiate the non-ISAKMP SA, IKE provides perfect forward secrecy of both keys and identities if:

- Main Mode is used to establish an ISAKMP SA.
- A Quick Mode exchange negotiates protection for another security protocol.
- The ISAKMP SA is deleted and not used for another exchange.

An internet draft for IKE version 2 [21] is being worked on which among other things combines many of the documents previously used to describe IKE functionality. The differences between the functionality of this new protocol and the current IKE protocol has not been examined in this project, but at the very least having previously separate protocols merged into a single document will make analysis and implementation of IKE easier.

### 8.2.3 MIKEY

‘Multimedia Internet KEYing’ [2] is a key management protocol developed specifically for use with real-time multimedia applications. The specification describes in detail its use with SRTP and support peer-to-peer scenarios as well as group scenarios.

MIKEY is meant to provide the same basic services as IKE with some added design goals specifically catering to the needs of real-time multimedia applications. The MIKEY protocol tries to be as simple as possible while being designed for low bandwidth consumption, low computational workload, a small code size and a minimal number of roundtrips.

Perhaps the most interesting feature of MIKEY in the context of securing the SIPcon1 application is the ability of MIKEY to be integrated in a session establishment protocol such as SDP. This is possible as MIKEY, unlike IKE, can perform its operations in only two roundtrips, making the offer/answer model of a SIP INVITE/OK message a perfect carrier. MIKEY also supports the bidding-down protection required by this integration, a mechanism that will be described further in section 9.3.2.

When a MIKEY exchange is executed the keying material exchanged is not the Traffic-Encrypting Key (TEK) to be used by the security protocol. Instead, the exchange serves to let the parties agree on a TEK Generation Key (or TGK), which can be used to generate the TEKs needed. This is done to allow MIKEY to provide keying material for an entire Crypto Session Bundle (CSB) with only a single exchange. A crypto session bundle is a collection of Crypto Sessions (CS) where each crypto session shares common parameters but requires different TEKs. For example, an SRTP stream and its corresponding SRTCP stream can use the same TEK and would in that case belong to the same CS, another pair of the same type of streams could use the same parameters, but not the same keys and could therefore belong to the same CSB, using the same TGK to generate individual TEKs.

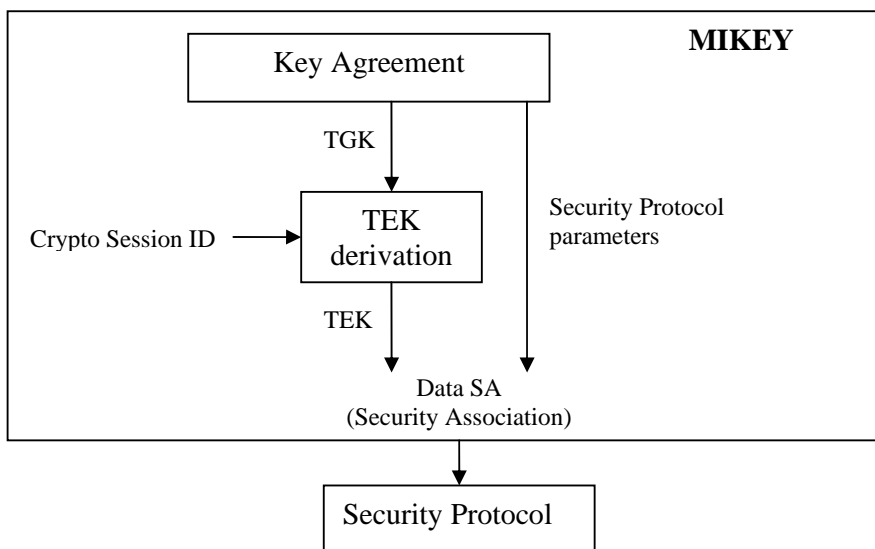


Figure 8.3: MIKEY overview

MIKEY employs three different kinds of key exchange mechanisms; pre-shared key, public-key encryption and Diffie-Hellman (with public-key encryption use to prevent man-in-the-middle attacks). The Diffie-Hellman mechanism is the only one optional to implement, but despite this and its higher resource consumption it is perhaps the most interesting since it is the only mechanism that provides perfect forward secrecy and allows the responder to influence the choice of keying material. Using this method for authentication, as in the case of using public-key cryptography to exchange the keys, introduces the need for a PKI.

There is a draft [11] that tries to limit the resource consumption of using Diffie-Hellman, and remove the need for a PKI, by authenticating the participants using a pre-shared key and keyed hash functions. This method is unfortunately not suitable for SIPcon1 purposes as it introduces the same problems regarding scalability as using the pre-shared key method.

Because identity protection would require additional cryptographic mechanisms the added complexity of using identity protection was seen as contradictory to the design goals of MIKEY, and thus identity protection was not included. It was also recognized that should the tunnelling feature of MIKEY be used, for example by integrating it in a SDP message, the exchange would inherit the flaws and security features for identity protection introduced by that messages' security mechanisms, or lack thereof [2].

## 9 Media Security Negotiation

When initiating a SIP call between two arbitrary clients there needs to be a way in which the cryptographic parameters of the resulting session is negotiated. There needs to be agreement over which crypto-suite to use, and what to set any crypto-suite specific parameter values to. Some encryption protocols, such as IPsec and TLS, provide mechanisms for the exchange of keys as a part of the protocol whilst others, most notably SRTP, do not. Since SRTP is considered to be the most appropriate way to encrypt real-time media it is important to find a practical way of negotiating the needed parameters and keying material.

### 9.1 Independent Key Management

Any applications communicating with each other can establish keying information out of band using some predetermined key management, as long as any firewalls between them allow the communication and both participants support that key management protocol. This still suggests the applications using this method have already determined what key management to use, when and how to use the key management. This is obviously not possible for any application that claims to promote interoperability with other applications and deployment in diverse environments which is why there needs to be some standardised way of exchanging cryptographic information.

Consider a scenario where two clients support a number of key management protocols. For each key management protocol a port would have to be monitored and the firewall set to allow traffic through that port. When key management communication is initiated it is performed unrelated to any SIP security and out of synch with the SIP negotiation. No part of the SIP offer or answer could be used to attach cryptographic relation between media streams (or give material to an entire session) and no result of the security negotiation would be present in the SIP response. In addition to this the key management queries would need to be sent sequentially in order of preference, each query sent after a delay which ensures a recipient supporting the protocol has time to respond. This approach would increase the time required to set up a session, and give an attacker with control over the flow of information the opportunity to choose the weakest key management supported by preventing any previous offers from reaching their destination.

### 9.2 The SDP Encryption Key Attribute

There currently exists one field in SDP to transport keys, the “k=” field [13]. However, this field is not sufficient for the purpose of establishing cryptographic parameters. Firstly, a great deal more than just the key needs to be transported. Secondly, there is no way to extend the field into several different alternatives and, thirdly, negotiating individual parameters for each media stream is not possible.

### 9.3 SDP Security Negotiation Protocols

There has recently been work done by the IETF on ways to negotiate security configurations in an SDP context. Two internet drafts being worked on to supply two different ways of negotiating such a security configuration are of particular interest; ‘Key Management extensions for SDP and RTSP’ [3] and ‘SDP Security Descriptions for Media Streams’ [1].

Adding security negotiation in the SDP body of a SIP message is convenient not only because of the textual format of SDP but also because there are already other reasons to secure SIP messages implying that the security negotiation might piggyback on the already established security mechanism for SIP.

### 9.3.1 SDP Security Descriptions for Media Streams

The ‘SDP Security Descriptions for Media Streams’ [1] document defines an SDP attribute to be used for encryption of unicast media streams. The attribute provides the necessary information for two applications to set up a common security association for the encryption of session media, specifically using SRTP.

The main consideration with using this new attribute is that the information exchange itself is not secure. The media security descriptions do not provide authenticated key establishment (AKE) services and all information is sent in plaintext. The media security description attribute provides nothing except a convenient format in which to communicate a desired security association. For this attribute to be used in a secure manner the SDP description itself must be somehow protected by an encapsulating data-security protocol such as IPsec, TLS or SIP S/MIME.

The crypto attribute contains four information fields; the tag field, the crypto-suite field, the key-params field and the session-params field. The tag field is filled with an identifier for that crypto attribute that must be unique among all crypto attributes for a given media line. The tag is used with the offer/answer model to determine which of the offered crypto suites were chosen by the answerer. The crypto-suite field is an identifier describing the encryption and authentication algorithms to be used for the media stream while the key parameters and session parameters fields are used to provide keying material and session specific parameters respectively.

When using “sdescriptions” to propose security for a media stream, crypto attributes are included below the media attribute, in the initial request, in the order of preference (first is the most preferred). Each attribute carries a unique tag, the crypto-suite proposed, a set of key parameters and, optionally, a set of session parameters.

The Recipient of the request generates an answer accepting exactly one of the crypto attributes, otherwise the offered stream must be rejected. The accepted attribute should be the first one supported, and hence the most preferred by the creator of the initial offer. A response is sent with the same tag and crypto-suite as the accepted offer together with the responder’s key information, declarative parameters and his response to any negotiable parameters.

If all mandatory parameters are supported on both sides the exchange of cryptographic information to fully initiate the media transfer is completed as soon as the offerer has received the answer to his request. Note that the answerer might start to send encrypted media even before the offerer receives the response to his initial request.

The “Security Descriptions” document [1] defines the use of crypto attributes in SDP when media streams are to be sent using SRTP. Use of the document with other transport protocols is allowed as long as the definitions within are extended to that protocol. For the keystream used by SRTP to be used correctly there are a couple of issues that needs to be addressed: Keystream re-use and key synchronization.

Because the ‘Security Descriptions’ protocol does not ensure session participants use different SSRCs, there is a slight chance that two participants might generate the same SSRC and thus cause an SSRC collision devastating for the security of the SRTP protocol if the same master key is used. The security descriptions document therefore requires each participant to use a unique master key (which is why the answerer needs to include his key information in the answer).

SRTP calculates index implicitly using a rollover counter (ROC) and a sequence number. To avoid synchronisation issues it is required that the ROC is set to zero at the start of the session (which restricts the specification to scenarios using unicast transmission) and that the random sequence number should be initiated to a number between 0 and  $2^{15}-1$ . The restraint on the sequence number is to avoid that the initial sequence number is close to wrapping (which otherwise could cause synchronisation issues at media stream initiation).

### 9.3.2 Key Management Extensions for SDP

This draft [3] describes how to negotiate cryptographic parameters by defining a new type of key management attribute. The attribute can be used to set up cryptographic parameters for a particular media stream or an entire session.

Each key management attribute includes the id of the key management protocol that will be used to generate that attribute’s message. The message for the key management field is generated by the appropriate key management protocol of the initiator and then base64 encoded whereby the message is encapsulated in the attribute. The result is a key management message that can be included in the SDP description thus negating the need for external key management signalling.

A receiver side implementation processes the attributes in a from top to bottom fashion, identifying which key management protocol has been used to create the message for that particular attribute (by examining its prtcl-id field) and, if the appropriate key management is available, sending the message, base64 decoded, to key management for further processing. In cases where an answer is required the key management of the first successfully processed and accepted key management message (for each stream/session) is used to create a response message which is then base64 encoded and sent in the response message.

The call initiator processes this answer exactly like the answerer did, passing it to key management and, if nothing goes awry, proceeds with the normal setup.

Including a number of possible offers in the SDP message regarding the encryption of a particular media stream (or for the entire session) ensures that a receiver that doesn’t support the initiators preferred offer still has a chance of setting up a secure context with any alternative key management protocols, cryptographic suites, or parameters suggested in the following offers. However, this feature can be exploited in what is known as a “bidding down” attack if the SDP description itself isn’t integrity protected. The bidding-down attack,

in this situation, is where a man-in-the-middle can remove cryptographically strong offers leaving only the weaker ones for the receiver to process. To ensure this weakness is not exploited it is required that the list of protocol identifiers is provided by the SDP application to each of the key management protocols to be used (protocol ids in the same order as in the SDP description). Each key management used in a key management extension implementation of SDP must be able to (and specify how to) process this list to ensure its authenticity. By comparing the list with the received offers any discrepancies can be detected. It is important to understand that the security level of the authenticated protocol identifier list will only be as high as the weakest protocol, hence the protocols offered should not vary to greatly in levels of security. Example 9.1, taken from [3], demonstrates three offers of encryption to the session using MIKEY and two fictitious protocols called KEYP1 and KEYP2. The protocol list that would need to be verified by the key management protocols in this example would be “mikey;keyp1;keyp2”.

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=Secret discussion
t=0 0
c=IN IP4 lost.example.com
a=key-mgmt:mikey AQAfGM0XflABAAAAAAAAAAAAAAAAAsAyO...
a=key-mgmt:keyp1 727gkdOshsuiSDF9sdhsdKnD/dhsoSJokdo7eWD...
a=key-mgmt:keyp2 DfsnuiSDSh9sdh Kksd/dhsoddo7eOok727gWsJD...
m=audio 39000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 42000 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

Example 9.1: Demonstrates three offers to secure a session

It is also important to note that the supported key management attributes be sent all at once in the manner described above. This is important because sending back-to-back invites with a single, new cryptographic offer in each would open up the possibility of the bidding-down attack (by the man-in-the-middle ensuring no offer other than a weak one is accepted). The nature of bidding-down attacks makes them particularly dangerous as they provide a false sense of security for the involved parties. On the other hand an attacker that has performed a bidding-down attack still has to break the security of the least effective encryption to gain access to the information. One unavoidable weakness of the SDP key management extensions method is that an attacker is always able to peel of an entire set of security offers to produce an unencrypted session or stream (since the integrity protection is contained in the key management attributes the attacker has removed). As this attack results in an unencrypted session rather than weak encryption the user will at least be made aware that the session is not secured and has the choice of aborting the session. By not accepting non-secure sessions this attack would only result in denial of service, which could just as easily be achieved by hindering the SIP invite.

The SDP key management extension draft was constructed with the key management protocol MIKEY in mind. This means that MIKEY supports all necessary features for key management extensions to be successfully implemented. Other key management can be implemented as long as the protocol supports required features (Initiation requires no more

than one offer/answer, key management list authentication support implemented, etc.) and any particulars of such an implementation are documented.

The requirement for authentication imposed by the use of MIKEY in this way requires that user agents be able to authenticate themselves to each other. Implicitly this means that there needs to be a PKI available to handle certificates for the end-to-end authentication.

## 10 SIP Encryption

Securing SIP messages is vital for applications that supply keying material tunnelled through the SDP body of the message. Nevertheless, solutions which do not employ such tunnelling have immense use for encrypted SIP messaging as this prevents an attacker from controlling session parameters. The information in the SIP message essentially decides every aspect of the subsequent session. Sending these messages in plaintext without employing integrity protection gives an attacker not only information about how the session is set up, but also the opportunity to change the session parameters.

### 10.1 Hop-by-hop Security

The motivation for encryption of the SIP headers is as mentioned before the integrity, authenticity and confidentiality of the SIP message. This becomes especially important when SRTP is used to encrypt the media stream. As SRTP does not provide any key agreement some external means of accomplishing this is needed. The methods described using the SIP message body to tunnel this information more or less rely on the security of the SIP message to provide this information securely to the other participant.

Because SIP messages are routed by intermediate proxies that require access to some of the SIP header fields, encryption of these fields cannot be applied end-to-end. Message fields such as the Request-URI, Route, and Via header need to be visible to (and in some cases modified by) proxies for SIP requests to be routed correctly. This is why any security mechanism which ensures confidentiality or integrity over these SIP headers must be applied on a hop-by-hop basis. This means messages are decrypted at each hop then encrypted anew before being transmitted to the next hop.

With hop-by-hop security there is always the possibility that any flawed security policy or malicious proxy along the way might ruin the integrity and confidentiality of the data transfer. Even though an inbound proxy could be configured to ignore requests from certain domains, or even ignore all requests except those from explicitly trusted domains, an application designed for interoperability should consider using some form of end-to-end security to avoid clumsy and restrictive ad-hoc proxy trust configurations.

#### 10.1.1 SIPS using TLS

TLS [9] is, as mentioned before, a security protocol that can be run over connection-oriented transport protocols such as TCP and supports any application protocols using TCP for the transfer of data. However, this means that a SIP message using TLS can not be sent over UDP as is the practice of the current version of the SIPcon1 application.

The recommended way to secure SIP message transfer in the Session Initiation Protocol is by using TLS. As mentioned before TLS is applied to any message bearing a SIPS URI instead of the usual SIP URI. Because a SIP proxy must be able to provide TLS encryption to any

message using a SIPS URI, providing TLS as a security measure between the UAC and its proxy can be convenient.

TLS is a protocol with many uses, but as such it is also quite an extensive protocol. Because of the way SIP requests are routed TLS cannot be used to encrypt an entire SIP message end-to-end. To be able to secure SIP with TLS every hop along the route must authenticate the next-hop (proxy server, target system), establish a secure TLS connection with the entity of that hop (although might have been done already at registration), and encrypt any SIP messages sent.

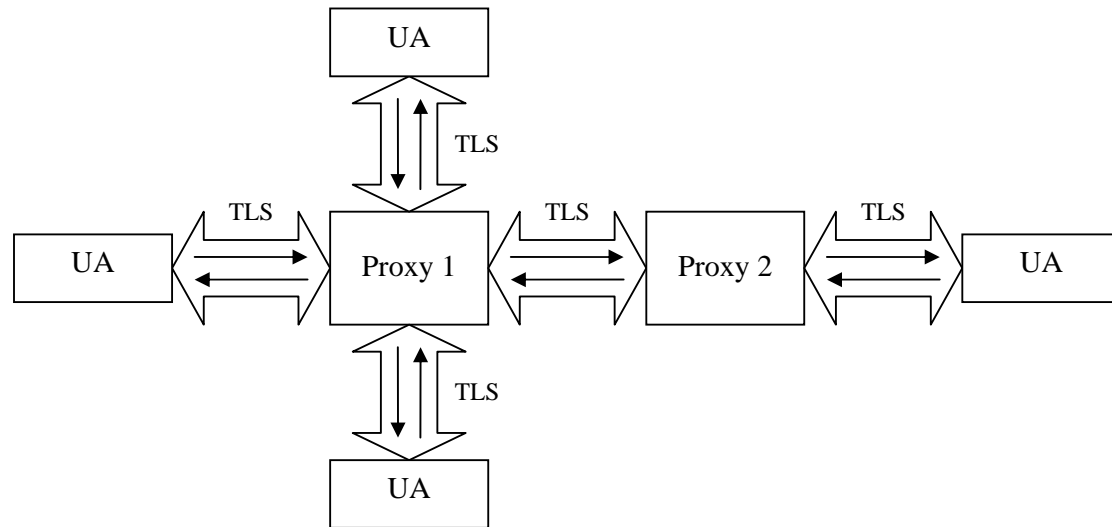


Figure 10.1: TLS connections for hop by hop security

The need to set up a security association between each hop and the use of TCP implies that a state needs to be maintained between the hops as the procedure of setting up a security association for each transaction, or even for each call, would produce too much overhead. It is also possible that any firewalls between a user client and a proxy relaying inbound calls might prevent the proxy from initiating a TLS connection to the UA, should a state between user client and inbound proxy not be maintained. It is therefore recommended that TLS connections created upon UA registration are maintained for the duration of the registration lifetime. In cases where the registrar, inbound and outbound proxy are different entities TLS connections should be set up to the proxies when registration has been completed.

One concern of using TLS in this way is that maintaining a connection between each client might not scale well to cases where many users are simultaneously connected to a proxy. The overloaded proxy would have a large amount of resources allocated to maintaining stateful TLS connections, many of which might not even be used at the time. There is also additional work associated with an implementation of TLS as a SIP application needs to be tightly coupled with TLS. In spite of these complications it is important to recognize that, because of the number of applications that can use TLS, any company wishing to secure SIP content might have additional use for a TLS implementation or might even already use one for another purpose.

Because transport mechanisms in SIP are specified on a hop-by-hop basis there is no assurance that TLS, when using a SIPS contact URI, will be used end-to-end as this is done at the discretion of the proxy servers. This dependency means that a proxy not following sound security policies might route a received SIPS request without TLS exposing the contents to any potential eavesdroppers.

Although the problem of authentication still leaves servers (proxy servers, redirect servers, registrars) depending on site certificates, either signed by a trusted CA (Certificate Authority) or signed by other providers with which communication is desired, authenticating a user to the server can be handled on a local level, using digest authentication [12] within the established TLS connection. Digest authentication parameters can be configured on installation of the application according to the security policy employed by that registrar. This means that only the servers need trouble themselves to obtain site certificates allowing large scale deployment without scalability issues.

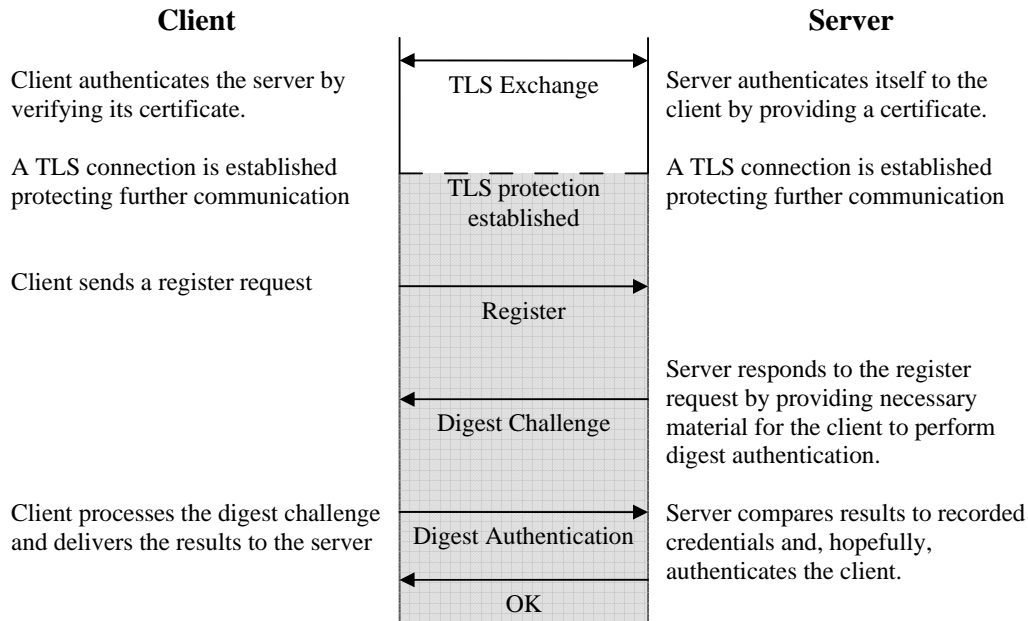


Figure 10.2: TLS single-authentication mode in concert with HTTP Digest

When coming online a UA establishes a TLS connection to its SIP registrar. The registrar provides its certificate to the UA which in turn checks and validates the certificate with root certificates held by the client. If the certificate checks out the establishment of the TLS connection signifies the UA has agreed to trust the registrar's certificate. To ensure the registrar can trust the UA is a legitimate user it responds to the register request sent after TLS establishment with a 401, "Proxy Authentication Required", message. The UA then resubmits its register request using a Proxy-Authorization header field and the digest credentials established with the registrar on an earlier occasion. For the SIPcon1 application this UA to registrar authentication secret consists of a password provided on application configuration. The requirement of user agent authentication prevents an attacker from impersonating a legitimate user by forging register requests. Using TLS to secure this communication prevents

an attacker from intercepting any register request information that could aid him in launching an attack (for an example a replay attack, although this is supposed to be protected from by other mechanisms, namely by the incremented nonce counter in the digest authentication).

TLS communication between a UA and its next-hop proxy server/servers, and TLS between proxy entities, when delivering SIP request messages works much like the registration process. An entity should have set up a connection to its next-hop when coming online and should use this connection for any communication going through that next-hop entity, the difference being that proxies can provide mutual TLS authentication rather than using TLS in concert with digest authentication. Mutual authentication using TLS is possible between clients and servers in architectures where a PKI is present. A PKI would allow client to provide their own certificates and negate the need for Digest authentication. UAs with certificates would receive the added benefit of being able to act as TLS servers, which means a server could initiate contact with UA if no TLS connection is present. Without mutual TLS authentication UAs are solely responsible to initiate the TLS connection.

Other than the obvious risks of hop-by-hop security, involving SIP messages appearing in plaintext at each hop, TLS also has another, more subtle, security flaw. An attacker with access to the flow of information can still see the TCP and IP header information of a packet secured with TLS. This implies that an attacker with the ability to monitor traffic can perform traffic analysis to log time and destination of the network traffic. Traffic analysis of SIP signalling should not be a great cause for concern as SIP messages are sent on a hop-by-hop basis. An attacker would at most be able to figure out who has contacted a proxy and when. Nonetheless, this monitoring is a legitimate concern, however small, and for some applications might be considered important.

### **10.1.2 SIP with IPsec**

Although a message containing a SIPS URI is supposed to be secured using TLS along the entire route IPsec may be used instead as a local policy between the proxy and client. The use of IPsec in this context could prove useful when the protocol is already used for other security purposes. The registration procedure when using IPsec would be similar to the one using TLS, only IPsec secured traffic would ensue instead of a TLS connection.

IPsec is most commonly used in VPN scenarios. Although the protocol can be used to prevent some forms of traffic analysis, other characteristics of IPsec and SIPS makes it less useful. First of the use of SIPS to indicate a secure SIP request implies the use of TLS and hence proxies would still need to use TLS to exchange messages. Anytime security becomes more complicated, by including a complex protocol or using many protocols, analysis of its level of security becomes more difficult and replacement of obsolete protocols becomes cumbersome. Complexity in this context is not a good thing. The complexity of implementing IPsec itself can be rather strenuous because SIP does not describe a framework for the use of IPsec. Also, no key management is specified for IPsec and IKE, the key management most commonly used with IPsec is a rather extensive protocol. Should the path to a proxy involve a NAT, additional problems may arise due to the NAT incompatibility of IPsec. The fact that IPsec is usually implemented on the operating system level adds further complexity.

There are no grand benefits of using IPsec to secure SIP requests, although as previously mentioned it does protect against traffic analysis. The biggest benefits of using IPsec do not stem from added security features as much as its overall usefulness (particularly in VPN

scenarios). Because IPsec can be used in many applications, supporting both UDP and TCP, and because it doesn't need to be as tightly coupled with the application as for example TLS it might be attractive in some instances to use IPsec because of practicality.

In short the benefits and drawbacks of using IPsec to secure SIP messages are similar to those discovered in the context of media encryption with a few major changes:

Less weight is added to the QoS issues as these delays only affect call initiation and configuration latency when SIP messages are encrypted with IPsec. In addition the problems of NAT traversal are not as likely to be prominent when a client connects with his proxy using IPsec as when a client uses it for end-to-end encryption to another client. In infrastructures where a proxy might provide services for clients physically distant from the proxy location any intervening network infrastructure could of course still cause a problem

Because of how SIP messages are routed the benefits reaped from tunnel mode in SIP signalling has limited use since messages are already sent hop-by-hop instead of end-to-end. An attacker performing traffic analysis on the traffic between the caller and the proxy would only learn the proxy is the destination of the request, not to which destination that request is to be forwarded. Similarly, an intercepted request between a proxy and the intended recipient does not yield information on the originator of the message.

### 10.2 End-to-end Security

As previously mentioned relying solely on hop-by-hop security makes it difficult to know for a fact the SIP and SDP information is kept confidential and unchanged throughout the entire request path. Depending on the application and its utilization a list of trusted or distrusted proxies can be maintained, although this method is rather crude and does not scale well. The reason why this method is not totally safe is that even if it can be guaranteed the providers of a proxy have no malicious intent it can never be guaranteed their security is infallible. In the end any end system can be compromised, and therefore security can never be guaranteed unless it is certain a system is free of malicious software and unreachable by any potential attacker either physically or by electronic means. Implementing end-to-end security at least allows a user the comfort of knowing that unless any of the end systems are compromised, either by running malicious software or being physically exposed to an attacker, the communication will be secure.

In an application where the only part imperative to protect is the media transfer, end-to-end security might entail applying end-to-end security on only the cryptographic information. This would allow an attacker with total access to a proxy relaying the message the opportunity to modify and change everything but the cryptographic information. Since this would allow a user to be certain any accepted cryptographic media streams are secure, the user could trust everything sent in the media stream will be fully protected. Of course, an attacker could peel of streams altogether, perhaps removing a video stream, or removing all offered codecs except one with inferior quality or heavier computational costs. Since, in the case of applied hop-by-hop security, this possibility would mean an attacker has full access to a trusted proxy the attacker could just as easily perform DoS on all requested messages by removing them completely, which suggests this risk might be acceptable. Attacks removing media streams, completely blocking traffic or in other ways modifying SDP data are bound to eventually be

noticed and dealt with. Falsely believing a compromised communication is safe, however, can go unnoticed for a long time in addition to being much more dangerous.

### 10.2.1 S/MIME

As SIP messages carry MIME (Multipurpose Internet Mail Extensions) bodies which means the standards for securing MIME contents, S/MIME [35] (Secure/Multipurpose Internet Mail Extensions), can be used to ensure the confidentiality and integrity of the SIP body end-to-end. This might constitute a problem for any of the less common network intermediaries requiring information from the SDP body to function correctly (certain rare types of firewalls).

To sign and encrypt the S/MIME bodies public and private keys are used. The use of S/MIME within SIP can provide authentication, integrity and confidentiality of signalling data. As access to a consolidated certificate authority to provide certificates for end-user applications is needed, a widespread and dynamic deployment of this solution relies on the existence of such an authority. Users should acquire certificates from known public certificate authorities, although because no authorities are available that provides certificates on a per-user basis there needs to be some other way to supply end users with certificates. This problem could be solved by having the proxies, which either way need some type of certificate for inter-domain authentication (such as TLS authentication) maintain a PKI (Public Key Infrastructure) based on that certificate.

Because of the problem of needing a central certificate authority users could create self-signed certificates. This however opens up the possibility for a man-in-the middle attack where an attacker intercepts the first exchange of keys between the two parties in a dialog and replaces the existing CMS-detached signatures in the request and response with his own certificate. As this can only be done on the first exchange of keys between two parties and because it might be difficult staying in the path of all future dialogs between the two parties this might not be an easy attack to launch. It is important to remember though that if hop-by-hop security is also utilized the only justification for using end-to-end security would be if one expects that a hop along the way could be compromised. This scenario would give an attacker the opportunities needed to launch this kind of man-in-the-middle attack.

There are requirements that when a UAS (User Agent Server) receives an S/MIME protected message that cannot be verified (which is the case with self signed certificates) the user should be notified and queried on whether to accept the message or not. While this would introduce the need to enable communication between the UAS and the user the bigger problem might actually lie in the fact that the user needs to be involved. Security measures utilized by an average user with no special skills in the area of cryptography should be able to operate transparently. The maximum user involvement in security issues for a user should be whether to accept a call where the security setup has failed or not, without having to distinguish between different levels of security. An application using this method might wish to present the call as having failed to set up any security at all to a user when the certificate can not be verified to spare the user of having to make the distinction between a completely secure and possibly compromised negotiation.

A user that sends a request using S/MIME and needs to communicate with the recipient must resend without S/MIME if a response is received that indicates the intended recipient doesn't support S/MIME. Like any mechanism agreeing on security by multiple requests this method

is susceptible to downgrade attacks. One could then argue that if an attacker with the ability to penetrate the hop-by-hop security can avoid S/MIME protection by simply not delivering the initial request there would not be any point in implementing S/MIME in the first place in an implementation that allows multiple requests. However, allowing end-to-end encryption as an added option when making a call guarantees that a successful offer is protected from compromised hop-by-hop security. Should this extra security measure be required a failure would of course constitute a failure to set up the session.

There is a means of providing some integrity protection for SIP headers using S/MIME. Described in the SIP specification [37], this is accomplished by encapsulating entire SIP messages within MIME bodies of the “message/sip” type and applying the MIME security in the same manner as in the case of securing SIP bodies. An unencrypted set of header fields would then be added to be compared with the encrypted fields at the destination to ensure message integrity. A more specific mechanism to achieve integrity and authentication using a similar method is presented in the ‘SIP Authenticated Identity Body Format’ [33] document. The AIB format allows the same functionality as the encapsulation approach previously mentioned (for the purpose of ascertaining caller identity) without including superfluous header information and thus decreasing the burden otherwise laid on the UAS to distinguish legitimate header changes from harmful ones.

### **10.2.2 Public Key Infrastructures**

For a security mechanism to work two participants need to have access to a shared secret or the other participant’s public key. This keying information can be exchanged through previously secured channels, but unless these channels have some way of being established without piggybacking on some other security this becomes an unsolvable loop. For example, digest authentication relies on the user having a shared secret with the registrar or proxy while TLS uses a certificate either obtained by a trusted CA (Certificate Authority) or previously exchanged with other trusted proxies according to some previously decided upon security agreement.

While the method of signing each other’s proxy certificates on an ad-hoc basis would work in a small scale scenario it does not scale well and cannot be deployed in a dynamic environment. Although using a commercial CA would present an additional expense it is a preferable alternative when considering interoperability and scalability.

One certificate for the proxy is enough to secure SIP communication with hop-by-hop security since the domain proxy can use the certificate to authenticate itself to other proxies and use digest authentication on registration to establish TLS connections with all users. By extension, a properly registered client with an unbroken and secure chain of TLS connections to the end destination ensures the originator of any requests sent through that path is indeed the authorized user (or at least the authorized device). As mentioned before though, transitive trust through hop-by-hop security has its weaknesses. To be able to provide end-to-end security every user needs some type of certificate in order to authenticate him- or herself to other users.

The problem with certificates is that in a large scale scenario, providing a personal certificate for every user is not feasible. This is why a Public Key Infrastructure, or PKI, is sometimes needed. By having a certificate issued by a CA a domain could issue personal certificates to its users based on the certificate provided by the CA. This would not be the equivalent of a

personal certificate since it would only be valid in the context of that domain or provider. If a user was to occasionally register with different domains one certificate would have to be acquired for each domain signed by the certificate of that domain.

Despite being advocated by SIP, S/MIME has not yet been widely deployed. A newly conceived IETF draft [20] attributes this to the complexity of providing a reasonable certificate distribution infrastructure. The document is meant to be able to provide certificates strongly binding the user's identity to the certificate without actually involving a Certificate Authority. If this mechanism proves effective it would be a very interesting alternative to solving some of the problems associated with certificate management.

### **10.2.3 Enhancements for Authenticated Identity Management in SIP**

Because of the difficulties associated with providing end users with legitimate certificates that can be trusted by a third party, many applications avoid security mechanisms which require a PKI. This means many SIP user agents lack the PKI needed to authenticate end users to each other using protocols such as S/MIME or TLS.

The ietf internet-draft 'Enhancements for Authenticated Identity Management in the Session Initiation Protocol' [34] is a newly conceived work-in-progress document which is aimed at providing end-to-end authentication in SIP by introducing a way to distribute cryptographically secure authenticated identities. Although too fresh to give a good indication of its potential usefulness the protocol has some good potential. The progress of the document's development could be of interest to anyone involved in SIP security.

## 11 SIP Security Negotiation

As previous analysis of required security has shown the SIP signalling traffic needs to be secured for numerous reasons. The two prevalent ones being to prevent session content from leaking to eavesdroppers and preventing attacker modification of SIP or SDP content. Before providing encryption, authentication and integrity protection of the SIP signalling traffic there needs to be a shared method available to do so. How to select the method to provide security services to a SIP session can be decided with a couple of approaches: Using a default mechanism or negotiating a mechanism.

### 11.1 Default Session Security

One common way of determining local security policy on SIP traffic is to simply determine which protocol to use beforehand. This is possible in uniform environments where, for example, a company only uses one type of SIP application or all SIP applications are known to support the same security mechanism. An application could also try to contact the intended recipient using all supported mechanisms one by one until successful. This approach is vulnerable to the downgrade attack and should only be attempted with offers that are of cryptographically equal safety as an attacker with the right access to the communication would always be able force an application to use the least secure mechanism.

### 11.2 Security Mechanism Agreement for SIP

Because it is a goal of most commercial application to be deployable in as many environments as possible an application with this goal as a high priority might wish to negotiate which security mechanism to use when securing the SIP transmission. Protocols for this kind of negotiation are important because sending multiple requests by using different protocols until one is agreed upon is vulnerable to a man-in-the-middle attack where an attacker can remove cryptographically secure offers until one more vulnerable to an attack is chosen. A negotiation protocol also paves way for the introduction of new algorithms and changes in the intended use for SIP (different security mechanisms might suit different scenarios).

A standard for this negotiation has been defined in “Security Mechanism Agreement for the Session Initiation Protocol (SIP)” [4] that allows a UA and its next-hop SIP entity to negotiate a security mechanism for the SIP traffic. Problems with using this kind of protocol can arise if the protocol is not widely implemented. If an application using the protocol fails to have the negotiation acknowledged by the intended recipient it can do one of two things: proceed with the most likely supported protocol or abort the request. The first response would leave the application vulnerable to downgrade attacks, negating the functionality of the negotiation protocol completely when such an attack is launched. An attacker could simply prevent the negotiation request from reaching its destination to incur this condition. The second response could possibly restrict interoperability, defeating the intended purpose of the protocol, by failing a request where the most likely supported protocol would otherwise have been supported by both parties.

## **IV Results and Conclusions**

## 12 SRTP Implementation

As previous examination of the problem of media encryption has shown SRTP seems to be the most suitable security protocol for securing RFC 2793 [16] payloads. Because the choice of this protocol to secure the media stream is expected for most scenarios an SRTP implementation has been constructed as a part of this project. The implementation, being made to suit the SIPcon1 application, was made in java based on the RTP implementation made by Omnitor.

Because the actual implementation of SRTP was not considered a high priority of the project objectives the implementation is not built to accommodate multiple ciphersuites and key-sizes. Although further development of the code could allow for these options the current code is built to support the default values and suites as specified in SRTP [6].

Although the authentication of SRTP packets is defined only as a recommended feature the importance of packet authentication is so great all SRTP implementations should support this feature. Being such an important feature authentication of SRTP packets warranted implementation.

The MKI was not considered vital to operation of VoIP type applications since calls are expected to be completed after a relatively short amount of time, not lasting the months or years that would make the MKI strictly necessary. Because the handling of the MKI information depends on the key management protocol and this implementation was meant to be made relatively independent of external protocols this feature was left out. Of course, to promote interoperability the implementation takes into account the possibility of the inclusion of an MKI in incoming packets and can in those cases send MKI fields, although after being parsed this information is blatantly ignored by the application and all MKI fields sent consist of a field of zeros.

Since authentication of SRTP is included adding replay protection is a rather simple matter. By having a bitmap keep track of which packets have been received counting window size packets back from the highest sequence number received replay protection is realized. Any packets higher than the highest sequence have naturally not yet been received while any outside the windows size are considered to old and are discarded.

Another optional feature allows for the refreshing of session keys whenever sequence numbers reach a number divisible by `key_derivation_rate`, where `key_derivation_rate` is a value negotiated as a part of the security parameters. Since session keys become less secure the more they are used for encryption the inclusion of this feature means that keys are harder to brake and that any leaked or broken keys expose at most `key_derivation_rate` number of packets. While this feature has been included in the SRTP implementation there are parts of the algorithm that could stand some improvement. Mainly, the problem lies with the fact that a new set of keys needs to be generated during communication which means added latency of packets whenever they reach a sequence number which requires a new key to be generated. This could be solved by pre-generation of keys by a separate thread so that a new set of keys are readily available before the new packet arrives. As the purpose of this project is mainly

geared towards providing protection for text media speed is not as critically important as it would otherwise be. Adding threads for generation of keys would add complexity to the program while not greatly benefiting the goals of the project and therefore this is left to future development. Another interesting possible impact on the QoS comes from the possible reordering of packets. As the current implementation only saves the keys of the highest sequence number a new set of keys must be generated for any packet belonging to an older set of keys (providing it has not been replayed). In theory this could mean that a new packet, requiring new keys, followed by two older packets would prompt three sets of key generations back to back; One for the new set of keys that are consequently saved, and two for the old packets. This drawback could in the future be amended by saving older keys so that retrieval becomes much faster.

The use of reoccurring key derivation might be especially motivated when used for encrypting text packets with redundant data. As the nature of redundancy forces repetition of previously sent text in a predictable pattern (due to predictable headers and repeated payloads) an attacker might gain some advantage when trying to break the encryption. Although this information ought to give only marginal benefits in an attack which would probably require vast amounts of computation the use of reoccurring key derivation would serve to counteract any weakness caused by the nature of the text packets.

To ensure correctness in the implemented algorithms the implementation has been tested with known test vectors. The key derivation function, the AES cipher in counter mode and HMAC-SHA-1 has been proven to comply with some known test vectors [6, 8, 18].

## 13 Security Scenarios

Securing communication always means additional processing of the data, which inevitably leads to computational costs. Also, any additional messages or packet extensions needed for the mechanisms serve to increase network congestion. The performance drawbacks of adding security features proves the importance of choosing a security configuration achieving needed goals while not proving too detrimental to QoS considerations.

By examining the goals which the security implementation is meant to achieve a solution might be chosen that achieves those goals without putting unwanted strain on application performance. Depending on the level of security needed for the application three different security levels, suitable for different scenarios, will be suggested. Suggestions are made for an application which, like SIPcon1, uses SIP to manage sessions and RFC 2793 payloads. These suggestions, shown in Figure 13.1, will be discussed in a way which clarifies the benefits and drawbacks of using that solution.

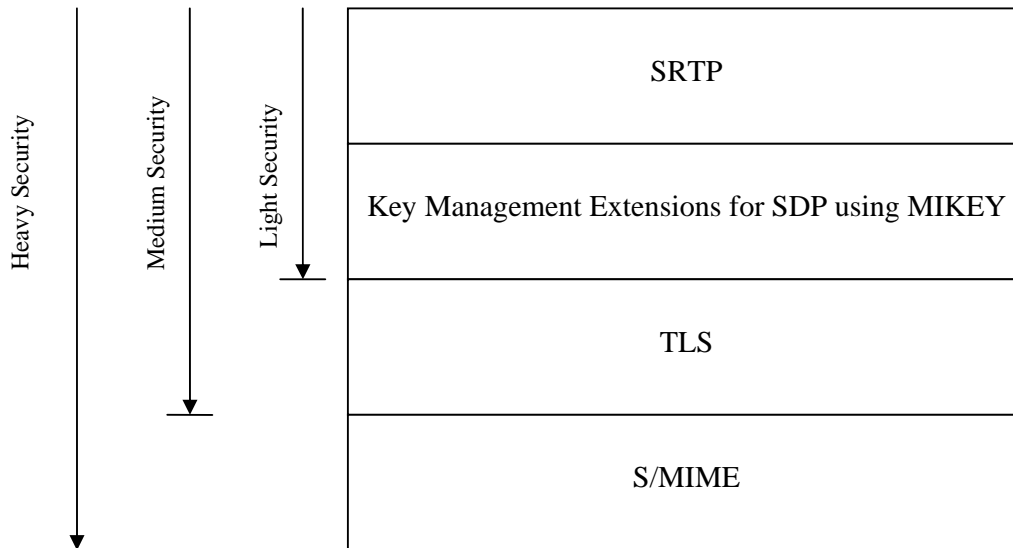


Figure 13.1: Security Layers in example scenarios

In addition to the security scenarios described in subsequent sections thought should be put into how to realize previously described non-software security aspects such as infrastructure, user education, good security policies and so forth. Some security features attainable by software implementations have more subtle uses than providing encryption (for example promoting interoperability) and can be considered for applications with varying demands on the level of security. For example, this is the case with the previously described negotiation

protocol for SIP security [4] which can be used to endow SIP hop-by-hop security with more flexibility.

### 13.1 Light Security

For some applications there is interest in ensuring secure communication with emphasis on maintaining a low computational cost and little overhead. Such concern might stem from limitations of the system on which the application is run or limitations in the network throughput. Also, the requirements of an application might simply demand low latency and jitter of packets.

As the encryption of the media is best accomplished by SRTP we will begin by choosing this protocol. By using SRTP not only are the RFC 2793 payloads handled suitably, but extension of the encryption to cover other types of real-time media is possible. Using SRTP adds the requirement of some type of key agreement since the SRTP protocol leaves this task to external protocols.

To conveniently provide keying information through the use of SIP and SDP two methods have been described. The security descriptions method of tunnelling information through SDP relies on the security of the SIP messages. Though securing SIP messages is a vital part of the application security, the premise for this level of security is a scenario in which the costs of added security features need to be low. Since adding security to SIP messages is costly we wish to explore an alternative in which this measure is not strictly necessary. The other approach to piggybacking the keying information, the key management extensions [3] described earlier, involves using MIKEY in a Diffie-Hellman exchange. Since MIKEY provides a method of protecting the integrity of offered keying material, and since the Diffie-Hellman method can be safely executed in the format of a MIKEY message, it is possible to tunnel this information in the SDP body without the risk of exposing secret key material.

Using the described method, exchange of the necessary information can be accomplished with very low computational and bandwidth cost. When using only the key management extensions, and consequently MIKEY, in this manner one has to be aware that no other aspect of the SIP information is secured. Once a secure session has been established one can be certain that the resulting media stream is secure, provided the security measures applied so far have not been cracked. There are, however, no guarantees that the attributes of the secured media stream (codec, port, sample rate, etc.) is unaltered.

Not using any authentication by way of SIP offers attackers some other options besides manipulating SIP invites and responses. Most notably, registration messages would not be secured either. Were configurations such as this to be employed where registration is performed using only user-supplied digest authentications attackers could impersonate registrars in the manner previously described since they would provide no means of authentication. Since a PKI would be needed for authentication with MIKEY said PKI could be used to employ some means of mutual authentication.

Although the security configuration just described is relatively fast, and does ensure any successfully established media streams are protected, it leaves an attacker many other areas to attack and manipulate. It is not recommended to use this type of security configuration as it

most likely would serve to induce a false sense of confidence in the security rather than offering robust protection.

### 13.2 Medium Security

Consider a scenario where an application is run on a reasonably powerful system with communication running over a reasonably effective and safe network. We want a security configuration which would be ample for most regular business calls or private conversations under normal circumstances.

As before, we will choose to use SRTP for the media encryption, which will again induce the need for some type of key agreement. Again examining the options we have for tunnelling within the SDP body of a SIP message we have two options on how to do this. Since we want a reasonable amount of security we will secure SIP messages. In doing so, we retain the option of using either the secure descriptions tunnelling or the key management extensions.

To secure the SIP communication we will choose TLS. This choice is made because it is an amply working standard with no complications in most environments. Proxy servers today support TLS and it is, by virtue of its presentation in the SIP protocol, the standard way of securing SIP messages. By using TLS we receive secure registration and hop-by-hop security which provide transitive security to SIP messages.

Going back to the matter of providing keying material the security provided by TLS is also applied to this tunnelled information. If we choose to use the attribute described in security descriptions no further security mechanisms are used and exchange is carried out easily. This approach can be chosen if we are absolutely sure the transitive protection provided by TLS is enough. At any time any part of the request path is exposed, for whatever reason, the keying information is exposed. This makes all security measures worthless and, worse yet, a session indistinguishable from a secure session is established. By following the key management extensions protocol the need for a MIKEY implementation and an established PKI arises. The benefit of using this protocol would be that an amount of end-to-end security is still maintained over the keying information even if the transitive security of TLS is compromised. Hence, an attacker which succeeds in gaining control of the TLS protected information will still face exactly the same security as described in the 'Light Security' scenario of the previous section.

### 13.3 Heavy Security

A weakness of the medium security example proves to be that breaking the transitive protection offered to the SIP messages by TLS would allow an attacker control over all session parameters not including any keying information protected by MIKEY. If there exists a chance that any entity part of the SIP routing might be insecure an application demanding a greater level of security might wish to add end to end protection to the SDP body of the SIP message. Additionally, end to end security over SIP headers not needed for routing by intermediate entities might be applied.

Understandably, any part of a SIP message needed by an intermediate entity to properly execute message delivery protected end to end will disrupt message delivery. For this reason

entities that need access to the SIP body, or any header being tunnelled, will not be able to deliver messages if end to end protection is applied to the needed information.

Despite S/MIME being advocated in the SIP specification no guarantee is given to the fact that a responder will support S/MIME. What this means is that an attacker might try a downgrade attack by preventing correct delivery of an S/MIME protected SIP message. It would then be up to the initiator to either resend an offer without S/MIME protection or to abandon the session. It stands to reason that this restriction might make it prudent to implement S/MIME protection as an added option for conversations that require the added protection. By actively choosing S/MIME protection for a conversation the user can be better informed on its functionality beforehand. An informed user that needs a secure communication, for example to enable an important online business conference, would be more inclined to abandon a communication where the S/MIME request fails, especially if it is known the intended recipient supports the security feature. A user receiving a prompt during call initiation asking if an offer should be re-sent without S/MIME could stand a greater risk of accepting the less secure mode.

When adding S/MIME protection the benefits of using MIKEY to secure keying material end to end are dwarfed. However, leaving provisions for cases where responders do not support S/MIME by keeping this protection seems natural. Using this modular approach adding to the level of security simply means adding another security feature, which gives a number of advantages. An application which supports multiple security features could be adapted to function smoother in a specific environment while an application under development could be built by developing these security features sequentially. Notably, since the use of MIKEY to secure keying material end to end would require the establishment of a PKI added S/MIME security could make use of the same PKI.

While using S/MIME to protect a message end to end it is important to note that, as is characteristic when adding security mechanisms, message size and packet latency will be affected negatively. Since TLS would imply the use of TCP for packet delivery added packet size would not imply added risk because of packet fragmentation, although network congestion could affect the QoS negatively. Added latency likewise will not affect the level of security, but will degrade the QoS for SIP message delivery.

## 14 Conclusions

Depending on the application in question, some security features might be more important than others. When dealing with an application using SIP to setup real-time media sessions it is recommended that SRTP be used to secure the media. Using the levels of security described in the previous chapter as a guideline security features can be added in the order considered most suitable for that application.

It is advisable to use at least TLS in concert with SRTP to secure the communication of an application since this creates a secure channel for entire SIP messages. Using TLS not only secures messages like invites but also ensures a better registration process. In environments where the security of SIP entities cannot be ensured (most environments) it is considered good practice to secure the SIP content, or at least part of the SIP content, end to end. This would prevent a security breach in the hop-by-hop security from exposing those parts of the SIP message secured end to end. This feature is especially important for any key information to be used by SRTP as exposure of such data would render the media encryption useless.

The security scheme described in the high security scenario of the previous chapter should give ample security to any VoIP-like application using SIP while the medium security might suffice in environments where control over the transient security provided by all TLS secured SIP entities is total. Another noteworthy fact about supplying security for an application is that circumstances surrounding the actual software can render any amount of programmed security worthless. Steps must be taken to make sure the software is used appropriately and that hardware and external software needed for operation of the application is protected just as well as the application itself. Maintenance and evolution of security mechanisms and policies should also be considered a high priority for any system where security is considered important.

## 15 References and Bibliography

- [1] Andreassen, F., Baugher, M., Wing, D. (Feb, 2005). "Session Description Protocol Security Descriptions for Media Streams". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdescriptions-09.txt> (Work in progress) {Acc: 2005-04-26}
- [2] Arkko, J., Carrara, E., Lindholm, F., Norrman, K. (Aug, 2004). "MIKEY: Multimedia Internet KEYing". Internet Engineering Task Force, RFC 3830.
- [3] Arkko, J., Carrara, E., Lindholm, F., Näslund, M., Norrman, K. (Mar, 2005). "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-mmusic-kmgmt-ext-14.txt>, (Work in progress) {Acc: 2005-04-26}
- [4] Arkko, J., Torvinen, V., Camarillo, G., Niemi, A., Haukka, T. (Jan 2003). "Security Mechanism Agreement for the Session Initiation Protocol (SIP)". Internet Engineering Task Force, RFC 3329.
- [5] Atkinson, R. (Aug, 1995). "Security Architecture for the Internet Protocol". Internet Engineering Task Force, RFC 1825.
- [6] Baugher, M., McGrew, D., Näslund, M., Carrara, E., Norrman, K. (Mar, 2004). "The Secure Real-time Transport Protocol (SRTP)". Internet Engineering Task Force, RFC 3711.
- [7] Bellovin, S., Schiller, J., Kaufman, C. (Dec 2003). "Security Mechanisms for the Internet". Internet Engineering Task Force, RFC 3631.
- [8] Cheng, P., Glenn, R. (Sep, 1997). "Test Cases for HMAC-MD5 and HMAC-SHA-1". Internet Engineering Task Force, RFC 2202.
- [9] Dierks, T., Allen, C. (Jan, 1999). "The TLS Protocol Version 1.0". Internet Engineering Task Force, RFC 2246.
- [10] Dierks, T., Rescorla, E. (Apr, 2005). "The TLS Protocol Version 1.1". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc2246-bis-10.txt> (Work in progress) {Acc: 2005-04-26}
- [11] Euchner, M. (Apr, 2005). "HMAC-authenticated Diffie-Hellman for MIKEY". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-msec-mikey-dhhmac-11.txt> (Work in progress) {Acc: 2005-05-01}
- [12] Franks, J., et al (Jun, 1999). "HTTP Authentication: Basic and Digest Access Authentication". Internet Engineering Task Force, RFC 2617.

- [13] Handley, M., Jacobson, V. (Apr, 1998). "SDP: Session Description Protocol". Internet Engineering Task Force, RFC 2327.
- [14] Harkins, D., Carrel, D. (Nov, 1998). "The Internet Key Exchange (IKE)". Internet Engineering Task Force, RFC 2409.
- [15] Hellström, G. (Jan 1998). "ITU-T Recommendation T.140 – Text Conversation Protocol for Multimedia Application". National Post and Telecom Agency, ITU-T T.140.
- [16] Hellström, G. (May, 2000). "RTP Payload for Text Conversation". Internet Engineering Task Force, RFC 2793.
- [17] Hellström, G., Jones, P. (Aug, 2004). "RTP Payload for Text Conversation". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-avt-rfc2793bis-09.txt> (Work in progress) {Acc: 2005-04-26}
- [18] Housley, R. (Jan, 2004). "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)". Internet Engineering Task Force, RFC 3686.
- [19] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., Stenberg, M. (Jan, 2005). "UDP Encapsulation of IPsec ESP Packets". Internet Engineering Task Force, RFC 3948.
- [20] Jennings, C., Peterson, J. (Feb 19, 2005). "Certificate Management Service for The Session Initiation Protocol (SIP)". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-sipping-certs-01.txt>, (Work in progress) {Acc: 2005-04-26}
- [21] Kaufman, C. (Sep 23, 2004). "Internet Key Exchange (IKEv2) Protocol". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-17.txt> (Work in progress) {Acc: 2005-04-26}
- [22] Kent, S., Atkinson, R. (Nov, 1998). "IP Authentication Header". Internet Engineering Task Force, RFC 2402.
- [23] Kent, S., Atkinson, R. (Nov, 1998). "IP Encapsulating Security Payload (ESP)". Internet Engineering Task Force, RFC 2406.
- [24] Kent, S., Atkinson, R. (Nov, 1998). "Security Architecture for the Internet Protocol". Internet Engineering Task Force, RFC 2401.
- [25] Krawczyk, H. (Nov 30, 1995). "SKEME: A Versatile Secure Key Exchange Mechanism for Internet". <http://www.research.ibm.com/security/skeme.ps> {Acc: 2005-04-27}
- [26] Krawczyk, H., Bellare, M., Canetti, R. (Feb, 1997). "HMAC: Keyed-Hashing for Message Authentication". Internet Engineering Task Force, RFC 2104.

- [27] Kuhn, R.D., Walsh, T.J., Fries, S. (Jan 2005). "Security Considerations for Voice Over IP Systems". National Institute of Standards and Technology, SP 800-58.
- [28] Maughan, D., Schertler, M., Schneider, M., Turner, J. (Nov 1998). "Internet Security Association and Key Management Protocol (ISAKMP)". Internet Engineering Task Force, RFC 2408.
- [29] NIST (Apr 17, 1995). "Secure Hash Standard". National Institute of Standards and Technology, FIPS PUB 180-1.
- [30] NIST (Nov 26, 2001). "Specification for the Advanced Encryption Standard (AES)". National Institute of Standards and Technology, FIPS PUB 197.
- [31] Orman, H. (Nov, 1998). "The OAKLEY Key Determination Protocol". Internet Engineering Task Force, RFC 2412.
- [32] Perkins, C., et al (Sep, 1997). "RTP Payload for Redundant Audio Data". Internet Engineering Task Force, RFC 2198.
- [33] Peterson, J. (Sep, 2004). "Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format". Internet Engineering Task Force, RFC 3893.
- [34] Peterson, J., Jennings, C. (Feb 16, 2005). "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)". Internet Engineering Task Force, Internet Draft. <http://www.ietf.org/internet-drafts/draft-ietf-sip-identity-04.txt>, (Work in progress) {Acc: 2005-04-26}
- [35] Ramsdell, B. (Jun, 1999). "S/MIME Version 3 Message Specification". Internet Engineering Task Force, RFC 2633.
- [36] Rescorla, E. (Jun, 1999). Diffie-Hellman Key Agreement Method. Internet Engineering Task Force, RFC 2631.
- [37] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E. (Jun, 2002). "SIP: Session Initiation Protocol". Internet Engineering Task Force, RFC 3261.
- [38] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V. (Jul, 2003). "RTP: A Transport Protocol for Real-Time Applications". Internet Engineering Task Force, RFC 3550.
- [39] Thayer, R., Doraswamy, N., Glenn, R. (Nov 1998). "IP Security Document Roadmap". Internet Engineering Task Force, RFC 2411.

## Appendix A – Acronyms and Abbreviations

AES	Advanced Encryption Standard
AH	Authenticated Header
AKE	Authenticated Key Establishment
CA	Certificate Authority
CS	Crypto Session
CSB	Crypto Session Bundle
CODEC	Compression/Decompression
DNS	Domain Name System
DOI	Domain of Interpretation
DoS	Denial of Service
ESP	Encapsulating Security Payload
HMAC	Keyed-Hashing for Message Authentication
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	IP Security Protocol
ISAKMP	Internet Security Association and Key Management Protocol
JMF	Java Media Framework
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extensions
MIKEY	Multimedia Internet KEYing
MKI	Master Key Identifier
NAT	Network Address Translation
NIST	National Institute of Standards and Technology
PKI	Public Key Infrastructure
QoS	Quality of Service
RFC	Request For Comments
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
S/MIME	Secure/Multipurpose Internet Mail Extensions
SA	Security Association
SDP	Session Description Protocol
SHA-1	Secure Hash Algorithm 1
SIP	Session Initiation Protocol
SIPS	Secure SIP
SPI	Security Parameter Index
SRTCP	Secure Real-time Transport Control Protocol
SRTP	Secure Real-time Transport Protocol
SSRC	Synchronization Source
TCP	Transmission Control Protocol
TEK	Traffic Encryption Key
TGK	TEK Generation Key
TLS	Transport Layer Security
UA	User Agent

UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
XOR	Exclusive OR logical operation