Evaluation of VoIP Security for Mobile Devices

In the context of IMS

PRAJWOL KUMAR NAKARMI

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

KTH Royal Institute of Technology
Master's Programme in Security and Mobile Computing - NordSecMob
Communication Systems (CoS)

Prajwol Kumar Nakarmi
nakarmi@kth.se

Evaluation of VoIP Security for Mobile Devices

in the context of IMS

Master's Thesis
Stockholm, June 16, 2011

Host Supervisor:	Professor Gerald Q. Maguire Jr.(*maguire@kth.se*) Royal Institute of Technology
Home Supervisor:	Professor Antti Ylä-Jääski, (*antti.yla-jaaski@tkk.fi*) Aalto University School of Science
Instructor:	John Mattsson, (*john.mattsson@ericsson.com*) Ericsson Security Research

Abstract

KTH ROYAL INSTITUTE
OF TECHNOLOGY
Communication Systems (CoS)
Master's Programme in Security and Mobile Computing - NordSecMob

ABSTRACT OF
MASTER'S THESIS

Author: Prajwol Kumar Nakarmi

Title of thesis:
Evaluation of VoIP Security for Mobile Devices in the context of IMS

Date: June 16, 2011 **Pages:** 12 + 68

Supervisors: Professor Gerald Q. Maguire Jr.
Professor Antti Ylä-Jääski
Instructor: John Mattsson

Market research reports by In-Stat, Gartner, and the Swedish Post and Telecom Agency (PTS) reveal a growing worldwide demand for Voice over IP (VoIP) and smartphones. This trend is expected to continue over the coming years and there is wide scope for mobile VoIP solutions. Nevertheless, with this growth in VoIP adoption come challenges related with quality of service and security. Most consumer VoIP solution, even in PCs, analog telephony adapters, and home gateways, do not yet support media encryption and other forms of security. VoIP applications based on mobile platforms are even further behind in adopting media security due to a (mis-)perception of more limited resources. This thesis explores the alternatives and feasibility of achieving VoIP security for mobile devices in the realm of the IP Multimedia Subsystem (IMS).

Keywords: VoIP, smartphones, IMS, SIP, SRTP, MIKEY-TICKET, GBA
GBA Digest
Language: English

KTH KUNGLIGA TEKNISKA HÖGSKOLAN

SAMMANFATTNING

Författare:: Prajwol Kumar Nakarmi

Titeln på Avhandlingen:
Evaluation of VoIP Security for Mobile Devices in the context of IMS

Marknadsundersökningar från In-Stat, Gartner och Post- och telestyrelsen (PTS) visar på en växande global efterfrågan på Voice over IP (VoIP) och smartphones. Denna trend förväntas fortsätta under de kommande åren och det finns stort utrymme för mobila VoIP-lösningar. Men, med denna ökning av VoIP kommer utmaningar som rör tjänsternas kvalitet och säkerhet. De flesta VoIP-lösningar för konsumenter, i datorer, analog telefoni adaptrar och home gateways, stöder ännu inte mediakryptering och andra former av säkerhet. VoIP-applikationer baserade på mobila plattformar är ännu längre efter säkerhetsmässigt på grund av en (miss–)uppfattning om mer begränsade resurser. Denna uppsats undersöker alternativ och möjligheter att uppnå VoIP-säkerhet för mobila enheter inom IP Multimedia Subsystem (IMS).

Språk: Engelska

AALTO-YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Prajwol Kumar Nakarmi **Diplomityön Otsikko:** Evaluation of VoIP Security for Mobile Devices in the context of IMS
In-Statin, Gartnerin, ja Ruotsin posti -ja tietoliikenneviraston (PTS) markkinatutkimusraportit paljastavat kasvavan maailmanlaajuisen kysynnän Voice over IP (VoIP) ja älypuhelimille. Tämän trendin uskotaan jatkuvan seuraavien vuosien aikana, joten mobiili VoIP-ratkaisut tulevat yleistymään. Siitä huolimatta VoIP:in kasvuun liittyy haasteita, kuten palvelun laadun takaaminen ja tietoturva-asiat. Useimpien VoIP-ratkaisujen käyttö, PC:issä, analogisten puhelinten adaptereissa ja koti gatewayssa eivät vielä tue sisällön salausta, eikä muitakaan tietoturvan muotoja. VoIP-sovellukset, perustuen mobiilialustoihin, ovat sitäkin enemmän jäljessä sisällön tietoturvaratkaisujen käyttöönotossa, johtuen epätietoisuudesta resurssien riittävyydestä. Tämä työ tarkastelee mobiililaitteiden VoIP-tietoturvan eri vaihtoehtoja ja niiden käyttökelpoisuutta IP Multimedia Subsystem (IMS):in piirissä.
Kieli: Englanti

Acknowledgment

I owe my gratitude to Professor Gerald Q. Maguire Jr., who is my host supervisor, for guiding me all the way. His immense knowledge and experience with the subject matter have helped me in all the phases of this thesis work. I feel very lucky to have him as my supervisor who always finds time, admist his busy schedule, for students.

I thank my home supervisor, Professor Antti Ylä-Jääski, for the timely help and suggestions regarding my thesis.

I am grateful to John Mattsson, who is my industrial supervisor and author of the MIKEY-TICKET protocol, for making available his experience and knowledge of industry standards.

I would also like to thank Oscar Olsson, my colleague at Ericsson Research, for helping me during the implementation phase.

I am thankful to Ericsson Research for providing me with the equipments necessary to conduct the thesis work. I experienced a wonderful, friendly and intellectual working environment here.

I thank all the open source communities and forums who are responsible for my ever growing knowledge.

I want to express my love for my friends and family.

Stockholm, June 16, 2011

Prajwol Kumar Nakarmi

Contents

List of Tables

List of Figures

Abbreviations and Acronyms

3GPP	3rd Generation Partnership Project
ACK	Acknowledgment
AES	Advanced Encryption Standard
AOR	Address-of-Record
AV	Authentication Vector
B2BUA	Back-to-Back User Agent
BSF	Bootstrapping Server Function
CNAME	Canonical Name
CRC	Cyclic Redundancy Check
CS	Crypto Session
CSB	Crypto Session Bundle
CSRC	Contributing Source
DH	Diffie-Hellman
DTLS	Datagram Transport Layer Security
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GHz	Gigahertz
GUSS	GBA User Security Settings
HP	Hewlett-Packard
HSS	Home Subscriber System
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPsec	Internet Protocol Security
JNI	Java Native Interface
KDF	Key Derivation Function
KG	Keystream Generator
KMS	Key Management Service
MAA	Multimedia-Auth-Answer
MAR	Multimedia-Auth-Request
MB	Megabyte

MGCP	Media Gateway Control Protocol
MIKEY	Multimedia Internet KEYing
MitM	Man in the Middle
MKI	Master Key Identifier
MTU	Maximum Transmission Unit
NAF	Network Application Function
PC	Personal Computer
PDA	Personal Digital Assistant
PSTN	Public Switched Telephone Network
PT	Payload Type
PTS	Swedish Post and Telecom Agency
QoS	Quality of Service
RAM	Random Access Memory
RFC	Request for Comments
ROC	Rollover Counter
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
S/MIME	Secure/Multipurpose Internet Mail Extensions
SA	Security Association
SD	SIP Digest
SDES	SDP Security Description for Media Streams
SDES*	Source Description (**only in case of RTCP report*)
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLF	Subscriber Locator Function
SRTP	Secure Real-time Transport Protocol
SSRC	Synchronization Source
TCP	Transmission Control Protocol
TEK	Traffic-Encrypting Key
TGK	TEK Generation Key
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UE	User Equipment
UICC	Universal Integrated Circuit Card
URI	Uniform Resource Identifier
USA	United States of America
VoIP	Voice over IP

Chapter 1

Introduction

In reports published by the Swedish Post and Telecom Agency (PTS) [5], the Voice over IP (VoIP) market share in Nordic countries shows very fast growth, with the IP telephony market share reaching 20 % of all fixed telephony[1] in Sweden already by 2009. Another interesting development is that VoIP is spreading from the fixed-line world to the mobile world. While Nokia introduced a native Session Initiation Protocol (SIP) stack in its Symbian phones some time ago [6, 7], Android recently introduced a built in SIP stack (starting from Android version 2.3) [8]. The growth of smartphones and possibility of cost efficient communication have catalyzed the evolution of VoIP on mobile platforms. Reports published by the market research firm In-Stat [9] forecast huge worldwide adoption of smartphones and VoIP, specifically:

2012 more than half of cellular handset shipments in the United States of America (USA) will be smartphones

2013 VoIP penetration among businesses in the USA will reach 79 %

2014 mobile VoIP users will rise to nearly 139 million

2015 annual business mobile VoIP gateway revenues will soar past 6 billion U.S. dollars

2015 annual worldwide smartphone shipments will be nearly 1 billion and IP phone shipments will exceed 40 Million

However, as we move from traditional telephony to VoIP, we face the inherent security issues of IP based systems. The availability of tools, such as Wireshark [10], makes it easy to sniff and listen to the VoIP conversations if one connects to a suitable point in the network. Due to the wide availability of computers and such software tools, VoIP calls are more susceptible to eavesdropping compared to Public Switched Telephone Network (PSTN) calls[2]. This lack of security in VoIP seems to be very serious, especially since most consumer VoIP solutions do not yet support encryption. Although Skype [12]

[1] Fixed telephony referring to PSTN, ISDN, and broadband telephony.

[2] Even though the PSTN calls are easy to eavesdrop, the equipment required to connect to high capacity links carrying PSTN calls is not readily available as in the case of VoIP. [11]

has positively accessed its security [13], it is proprietary software and therefore its technology can not be used by others. In addition, because the source code is closed, it is not clear what security mechanisms are used or who has access to the encryption keys[3].

There have been several efforts to address the security in VoIP based systems, these will be discussed in chapter 2 and chapter 3. However VoIP security is still not common even in Personal Computer (PC) applications. Due to the (mis-)perception that mobile platforms have limited resources, the development of VoIP security for mobile applications has lagged behind that of PC applications. Thus far, we have not found full blown VoIP security being implemented in any open-source SIP based mobile applications.

This thesis project will be an opportunity to explore the alternatives and feasibility of achieving VoIP security in mobile devices. VoIP security in itself is a broad topic (as it includes signaling security, media security, guaranteeing Quality of Service (QoS), etc). To focus this thesis project the specific aspects of VoIP security that will be addressed are described in the next section.

1.1 Goals of Thesis

This thesis project is primarily focused on VoIP **media** security, with the following goals:

1. Evaluate alternatives for realizing VoIP media protection in mobile handsets with the focus on SIP used together with SRTP.
2. Integrate the key management protocol MIKEY-TICKET, the context of 3GPP IP Multimedia Subsystem (IMS), into the software which realizes the first goal.
3. Analyze possible solutions to use password based authentication, the context of 3GPP IMS, and make recommendations to extension of the current standard that uses a Universal Integrated Circuit Card(UICC).
4. Offer recommendations to those implementing VoIP applications on mobile handsets (particularly for handsets running Android) based on measurements and analysis of software which realizes the first, second, and third goals.

1.2 Contribution

This thesis work provides a prototype of secure VoIP mobile client complaint with IMS standards. According to the author's knowledge, this work is the first to integrate MIKEY-TICKET into a VoIP client. Also this thesis is the first reference implementation of an ongoing 3GPP study on using SIP digest based Generic Bootstrapping Architecture (GBA). The measurements presented

[3]Recently, a Russian researcher Efin Bushmanov has claimed to have reverse-engineered the Skype (`http://skype-open-source.blogspot.com/2011/06/skype-protocol-reverse-engineered.html`)

in this report are also the first regarding the current generation of mobile devices (specifically Android handsets).

1.3 Structure of the Report

The rest of the report is organized as - chapter 2 describes the necessary knowledge from the literature required to understand the technologies involved in this thesis. Chapter 3 summarizes other theses and publications relevant to this thesis. Chapter 4 presents the design decisions that have been made regarding various technologies in order to realize secure VoIP in mobile devices. Chapter 5 discusses the implementation details of our prototype. Chapter 6 presents the results of measurements made on the implementation. Chapter 7 concludes this report and suggest some further work. Appendix A presents the message flow between various components during a secure VoIP call.

Chapter 2

Background

This chapter presents some background information required to understand the works done in this thesis project. It introduces the protocols and technologies that are related to Voice over IP (VoIP) and security in VoIP. It starts by introducing the concept of VoIP. Then, it presents protocols related to signaling, media transfer, and key exchange. Finally it discusses the mechanism for establishing and using the subscriber authentication.

2.1 VoIP

Voice over IP (VoIP) is a technology for transmission of voice over packet-switched IP networks. It is also frequently referred to as IP telephony or Internet telephony. The basic idea behind VoIP is to transmit digitized samples of voice over a data network and replay them at the receiver. While the "V" in "VoIP" standards for "voice" it should be clear that the media could be audio, video, timed text, etc.; thus the general service is multimedia communication over an IP network.

While there is a cost for network connectivity, VoIP applications, such as Skype [12], Yahoo Messenger [14], Google Chat [15], etc. allow "free" calls to their users. In this context "free" refers to the marginal cost of making each call, thus there is no per call charge and no per minute cost for a call. Long distance phone calls via VoIP service providers, such as Jumblo [16], are also generally cheaper than via traditional telecommunications service providers. VoIP reduces the infrastructure cost because a single network is used to carry both voice and data, and the packets only need to be delivered when there is media content to be delivered (thus enabling statistical multiplexing of the links). With a sufficient quality Internet connection and a VoIP service provider, the user can receive and make calls from anywhere. Additionally, it is possible to integrate VoIP services with other systems such as email, conferencing, and so on. Hence, VoIP offers flexible communication options at low operational cost.

Some modes of operations for VoIP are [17]:

- PC-to-PC,
- Phone-to-Phone,
- PC-to-Phone,
- Phone-to-PC, and
- Network to Network.

Our study will address PC-to-PC calls in context of handsets. The term PC, here, will refer to both generic computers and smartphones, e.g. when a VoIP call is made between two Android phones, the model is still PC-to-PC rather than Phone-to-Phone. We will reserve the term Phone-to-Phone to be a call that involves two traditional telephones attached to a traditional telephone exchange or exchanges making a call via the Internet.

Some important technologies and protocols related to VoIP are:

- H.323 [18],
- Session Initiation Protocol (SIP) [19],
- Media Gateway Control Protocol (MGCP) [20], and
- Real-time Transport Protocol (RTP) [3].

The following sections describe the technologies of interest to this thesis project. Section 4.2 describes H.323 briefly. Because MGCP involves controlling telephony gateways it will not be referred to further in this document.

2.2 SIP

The Session Initiation Protocol (SIP) is a signaling protocol that is used for management of multimedia sessions. SIP was defined by the Internet Engineering Task Force (IETF) [21] and the latest version of its specification is RFC 3261 [19] [1].

SIP is a text-based application layer protocol and uses Uniform Resource Identifiers (URIs) (e.g. `sip:nakarmi@kth.se`) to address the caller and callee. Similar to HTTP, SIP works in request-response transaction model, i.e. a client request invokes a method in the server and the server sends back at-least one response. SIP is independent of the underlying transport layer and the transactional mechanism allows it to use unreliable transport protocols such as UDP [22] or reliable transport protocols such as TCP, T/TCP, TCP over TLS/SSL, etc..

[1] RFC 3261 has been updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, and 5922.

As described in RFC 3261, SIP supports the following five features for multimedia communications:

User location

know where to contact the callee

User availability

know if callee is available and willing to communicate

User capabilities

know which media formats to use

Session setup

establish the session for communication between caller and callee

Session management

modify or tear-down the ongoing session

Some of important SIP related terms that will be used in this document are:

Call

A communication session between peers.

Conference

Communication session between multiple participants.

Address-of-Record (AOR)

A SIP URI where the user might be available.

SIP Transaction

Comprises all messages from the first request sent from the client to the server up to a final response sent from the server to the client.

User Agent Client (UAC)

A logical entity that creates request. The role lasts for the duration of transaction.

User Agent Server (UAS)

A logical entity that responds to a request. The role lasts for the duration of transaction.

User Agent (UA)

A logical entity that can act as both UAC and UAS.

Proxy

Primarily serves the role of routing SIP requests and possibly responses; and if necessary rewrites specific parts of a request message before forwarding it.

Dialog

A peer-to-peer SIP relationship between two UAs that persists for some time.

Back-to-Back User Agent (B2BUA)

A concatenation of UAC and UAS at the same time. It receives the request as a UAS and in order to respond to that request it itself generates requests as a UAC.

A simple scenario of Alice making a call to Bob (using SIP) is illustrated in figure 2.1.

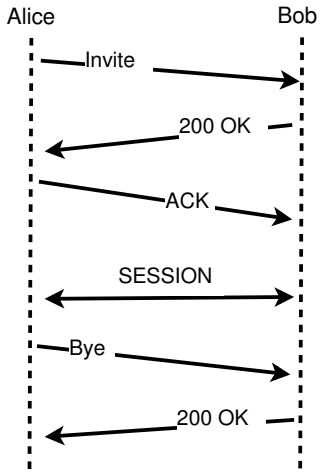


Figure 2.1: SIP Session Setup Example

Alice's INVITE message would look like the following (adapted from RFC 3261 [19]):

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

Similarly 200 OK message from Bob would look like the following (adapted from RFC 3261 [19]):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;
branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
```

```
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds;
received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

The SIP header fields have the following meanings:

Via

contains the address at which the client is expecting to receive responses. The branch identifies this transaction.

To

contains a display name and a SIP URI of the called party.

From

contains a display name and a SIP URI of the caller.

Call-ID

contains a globally unique identifier for this call.

CSeq

is command sequence that contains a sequence number and a method name.

Contact

contains a SIP URI that represents a direct route to contact the client.

Max-Forwards

limits the number of hops a request is allowed to traverse.

Content-Type

contains a description of the message body (not shown).

Content-Length

contains an octet (byte) count of the message body.

Unlike H.323, SIP is only involved in the signaling portion of a communication session. Thus SIP acts as a component which works with several other protocols to offer a complete multimedia architecture typically, using RTP for transporting real-time data (such as voice and video streams) and SDP for describing multimedia sessions in terms of protocols, port numbers, coder/decoders (CODECs), and so on.

The RFC 3261 [19] defines six request methods - REGISTER for registering

contact information, INVITE, ACK, and CANCEL for setting up sessions, BYE for terminating sessions, and OPTIONS for querying servers about their capabilities. Similarly, the specification allows six responses, as follow:

1xx(Provisional)

request is being processed

2xx(Success)

the request was processed successfully

3xx(Redirection)

further action needs to be taken for completing the request

4xx(Client Error)

bad request by client

5xx(Server Error)

request is valid, but server cannot fulfill the request

6xx(Global Failure)

the request cannot be fulfilled at any server

2.3 SDP

The Session Description Protocol (SDP) is a media description protocol which is intended for describing multimedia sessions. It provides a standard representation to convey session metadata such as transport addresses and media details. It is independent of the transport layer and does not handle the media encodings or the session negotiation by itself. The SDP standard was published and revised by IETF and is defined in RFC 4566 [23].

The presence of SDP is denoted by the media type *application/sdp* and a session description is composed of several lines with the following format:

```
<type>=<value>
```

The specification does not allow any whitespace between either side of "=" sign.

A simple SDP description is shown below (adapted from RFC 4566 [23]):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
m=audio 49232 RTP/AVP 98
a=rtpmap:98 L16/16000/2
```

The fields have following meaning:

v Protocol version. The specification describes it as 0

o Origin. The origin is specified in the format shown below:

```
<username> <sess-id> <sess-version> <nettype> <addrtype>
<unicast-address>
```

Note that there is a space, not newline (as above because it could not fit in single line), between <*addrtype*> and <*unicast-address*>. In the example given above, the user *jdoe* is using *IPv4 INternet* with address *10.47.16.5* with session-id *2890844526* and session-version *2890842807*.

s Session name.

i Session information. This is an optional field.

c Connection data. This is also an optional field. If present it has following format:

```
<nettype> <addrtype> <connection-address>
```

t Timing information in the decimal representation of NTP time values in seconds since 1900. It has the following format:

```
<start-time> <stop-time>
```

m Media description in the format:

```
<media> <port>/<number of ports> <proto> <fmt>
```

The examples above shows the audio data will use UDP port 49232 and the RTP protocol. The <*fmt*>format denotes the payload type numbers (the numeric value 98 is mapped by an *a* field - as described next).

a Media attribute. This is the primary means for tailoring SDP to particular media. The example shows the mapping of dynamic payload type 98 to 16-bit linear encoded stereo audio sampled at 16 kHz.

2.4 RTP

The Real-time Transport Protocol (RTP) is a transport protocol which provides end-to-end transport functions for delivering real-time data such as audio and video over IP networks. It is one of the technical foundations of VoIP and is used together with H.323 or SIP. Nevertheless, for RTP to be used in VoIP, there is a need for a separate signaling protocol, such as SIP and a media description protocol, such as SDP. RTP is defined by IETF [21] and the latest version of its specification is RFC 3550 [3].

The RTP header format is shown in figure 2.2. The first twelve octets are present in every RTP packet.

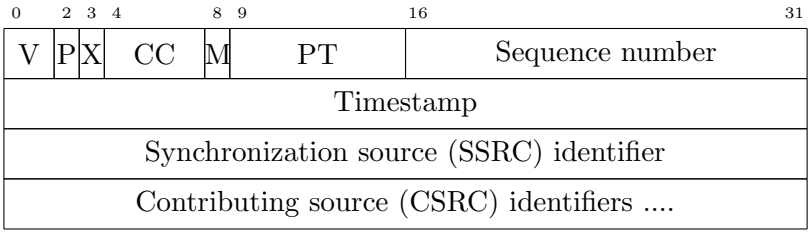


Figure 2.2: RTP Header Format [3]

The fields have the following meanings:

V

version number (current version is 2)

P

whether payload contains padding

X

whether header extension is present

CC

number of CSRC identifiers

M

profile specific marker for frames

PT

identifies the RTP payload type

sequence number

incremental numbers used to define a packet sequence and used by the receiver to detect packet loss

timestamp

sampling instant of the first octet in the RTP data packet

SSRC

identifies a synchronization source; this must be unique in each RTP session

CSRC list

list of contributing sources for the payload contained in the packet (0 to 15 sources)

Since real-time streaming applications require timely delivery of data and are resilient to some packet loss, RTP implementations are generally built on UDP. Therefore RTP does not guarantee any specific QoS for real-time services. However, as described in the specification, RTP can be augmented by a RTP Control Protocol (RTCP) to allow monitoring of the data delivery. While RTP carries data that has real-time properties, RTCP monitors the quality of service and conveys information about the participants in an on-going session. When

used together, RTP utilizes even port numbers and the corresponding RTCP stream uses the next higher odd port number. It should be noted that both RTP and RTCP are independent of the underlying transport and network layers.

If RTCP is used, then RTCP packets are periodically transmitted to all participants in the session in a similar fashion as the data packets. The RTCP packets carry the following:

SR

Sender report, for transmission and reception statistics from active senders

RR

Receiver report, for reception statistics from non active senders

SDES

Source description items, including CNAME

BYE

Indicates end of participation by a node

APP

Application-specific functions

According to the specification, RTCP performs the following four functions:

1. provide feedback on the quality of the data distribution,
2. carry a persistent transport-level identifier for an RTP source called the canonical name or CNAME,
3. control its packets in order for RTP to scale up to a large number of participants, and
4. convey minimal session control information, for example participant identification.

One of the main advantages of using RTP is that new multimedia formats can be added without changing the underlying standard. As such, application specific information are specified by RTP profiles and payload formats, and not included in the generic RTP header. For example, RFC 3551 [24] defines a set of static payload type assignments, and a mechanism for mapping between a payload format, and a payload type identifier using SDP. Another example of a profile that is relevant to this thesis project is SRTP [1] which provides a security service for RTP payload data.

2.5 SRTP

The Secure Real-time Transport Protocol (SRTP) is a RTP profile developed by researchers at Cisco and Ericsson. It was published by IETF as RFC 3711 [1]. SRTP was designed with the security goals of providing confidentiality, message authentication, and replay protection to the RTP traffic and RTCP. In addition to the security goals, SRTP has following properties that makes it a suitable

protection scheme in heterogeneous networks:

- low computational cost,
- low bandwidth cost (limited packet expansion, preservation of RTP header compression efficiency),
- small code size and data memory for keying information and replay lists, and
- independence from the underlying transport, network, and physical layers used by RTP, in particular high tolerance to packet loss and re-ordering.

The general idea is to intercept the RTP packets and to convert them to equivalent SRTP packets *before* sending them to the transport layer. The reverse is done on the receiving side. As such, SRTP can be considered a "bump in the stack". The companion protocol SRTCP (Secure RTCP) provides the same security services to RTCP as SRTP does to RTP.

SRTP uses cryptographic contexts - the information about the cryptographic state for sender and receiver. There are two types of keys: session keys and master keys. The session key is used for the actual encryption and message authentication; where as the master key is used to derive the session keys. It should be noted that the SRTP standard itself does not specify how to establish the master key. It is the responsibility of a key management protocol to determine the master key. MIKEY [2], SDES [25], and ZRTP [26] are examples of key management protocols.

The SRTP packet format is shown in figure 2.3. The Master Key Identifier (MKI) and the authentication tag are the only fields defined by SRTP that are not in RTP. The optional MKI field identifies the master key from which the session key(s) were derived that authenticate and/or encrypt this particular packet. The recommended authentication tag is used to carry message authentication data.

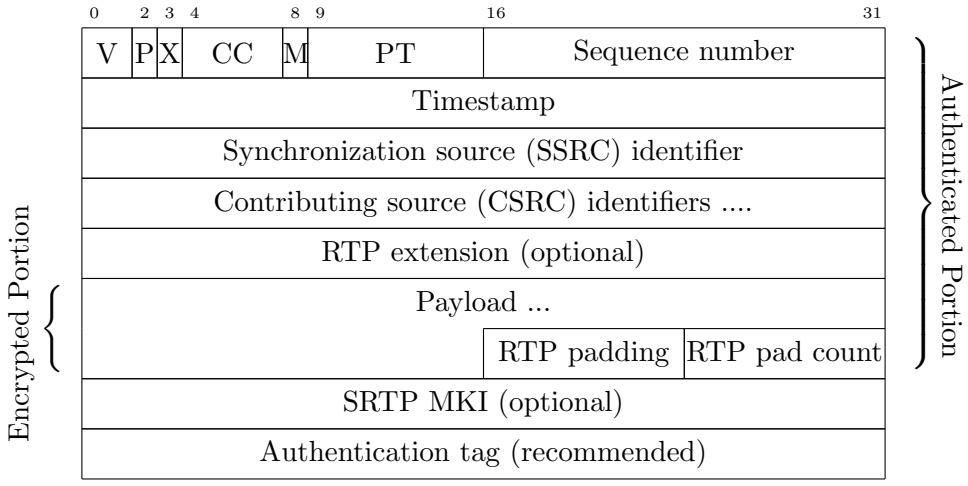


Figure 2.3: SRTP Packet Format [1]

Table 2.1 shows the default algorithms for encryption and authentications as defined in the specification.

Table 2.1: Encryption and Authentication Transforms in SRTP [1]

	mandatory-to-implement	optional	default
encryption	AES-CM, NULL	AES-f8	AES-CM
message integrity	HMAC-SHA1	-	HMAC-SHA1
key derivation (PRF)	AES-CM	-	AES-CM

The steps on sender side to construct the SRTP packet are:

1. Determine which cryptographic context to use (including which encryption algorithm to use)
2. Determine the index of the SRTP packet
3. Determine the master key and master salt
4. Determine the session keys and session salt
5. Encrypt the RTP payload
6. Append the MKI to the packet if required
7. Append the authentication tag to the packet if required
8. Update the Rollover Counter (ROC) [2] if necessary

Similarly the steps on the receiver side to optionally authenticate and decrypt the SRTP packet are:

1. Determine which cryptographic context to use
2. Get the index of the SRTP packet
3. Determine the master key and master salt
4. Determine the session keys and session salt
5. Check if the packet has been replayed and discard it if replayed
6. Verify the authentication tag and discard the packet if verification fails
7. Decrypt the encrypted RTP payload
8. Update the ROC and replay list
9. Remove the MKI and authentication tab if present

The process of encrypting the packet is shown in figure 2.4. It consists of generating a keystream segment corresponding to the packet, and then bitwise exclusive-oring that keystream segment with the payload of the RTP packet in order to produce the encrypted portion of the SRTP packet. Note that the keystream segments can be computed independently for each RTP packet and they can be computed in advance.

[2]The ROC used here is a 32-bit unsigned rollover counter. It records how many times the RTP sequence number has been reset to zero. It is used in determining the index of SRTP packet.

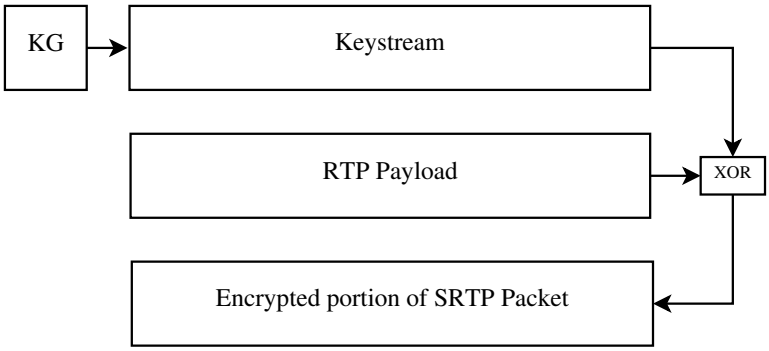


Figure 2.4: Default SRTP Encryption Process [1]

2.6 MIKEY

The Multimedia Internet KEYing (MIKEY) is a key management scheme that can be used with real-time applications. It is specifically designed to setup the encryption keys for SRTP-secured multimedia sessions. MIKEY was developed by researchers at Ericsson Research and the specification is defined in RFC 3830 [2].

MIKEY fits well in a heterogeneous environment because of following features:

1. Simplicity
2. End-to-end security only the participants involved in the communication have access to the generated key(s).
3. Efficiency in terms of:
   - low bandwidth consumption
   - low computational workload
   - small code size
   - minimal number of roundtrips
4. Tunneling, e.g. it is possible to integrate MIKEY with SDP.
5. Independence from any specific security functionality of the underlying transport

Some of the important definitions in MIKEY are listed below. These definitions can be related to SRTP as illustrated in table 2.2.

Data Security Association (SA) information for the security protocol like SRTP

Crypto Session (CS) data streams protected by a single instance of a security protocol e.g. RTP and RTCP can both be protected by single SRTP cryptographic context

Crypto Session Bundle (CSB) collection of one or more CSs

TEK Generation Key (TGK) a bit-string associated with CSB from which TEKs can be generated without needing further communication

Traffic-Encrypting Key (TEK) the actual key used to protect the CS

Salting key a random or pseudo-random string used to protect against attacks on the security protocol

Table 2.2: MIKEY-SRTP Relation [2]

MIKEY	SRTP
Crypto Session	SRTP stream (typically with related SRTCP stream)
Data SA	input to SRTPs crypto context
TEK	SRTP master key

MIKEY produces a Data SA to be used as input to the security protocol as shown in figure 2.5.

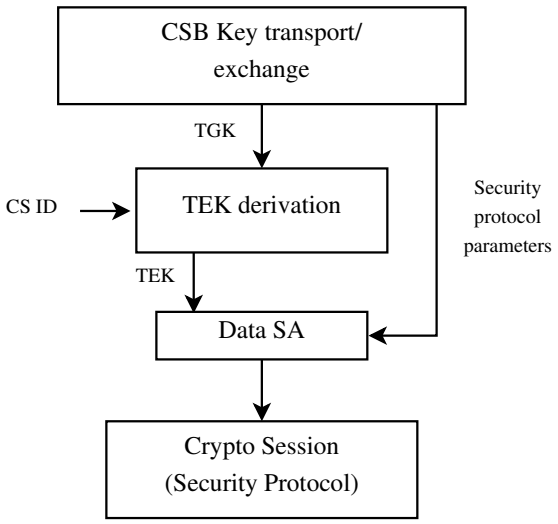


Figure 2.5: MIKEY Key Management Procedure [2]

The specification document for MIKEY specifies three methods for establishing a TGK:

Pre-shared key - uses symmetric cryptography and is most efficient to handle; however it is not scalable to very large numbers of user - although it may be feasible for a small to medium sized group of users

Public-key encryption - scalable but more resource consuming than pre-shared key

Diffie-Hellman key exchange - most resource consuming (in terms of computation and bandwidth), but provides perfect forward secrecy

2.7 MIKEY-TICKET

MIKEY-TICKET (defined in RFC 6043 [27]) is a key exchange protocol that extends MIKEY [2] with a set of new modes, all of which support the concept of a ticket similar to that in Kerberos [28]. MIKEY-TICKET uses a trusted Key Management Service (KMS) for ticket-based key distribution and is primarily designed to be used for media plane security in the IP Multimedia Subsystem (IMS), see section 2.11. IMS media plane security is discussed in section 3.13 on page 32.

MIKEY-TICKET requires up to three different round-trips as illustrated in figure 2.6. We assume that the KMS has pre-established trust with both the Initiator and the Responder. The KMS is involved only during the exchange of MIKEY messages and is not involved at all in securing the media session. The Ticket Request round-trip is used by the Initiator to request keys and ticket from KMS; the Ticket Transfer round-trip transfers a ticket to the Responder, and the Ticket Resolve round-trip is used by the Responder to request the keys mentioned in the ticket from KMS. The Ticket Request and Ticket Resolve round-trips can use either the Pre-Shared Key (PSK) method or Public-Key (PK) method of MIKEY. The RFC 6043 describes the four modes of MIKEY-TICKET operation as illustrated in table 2.3.

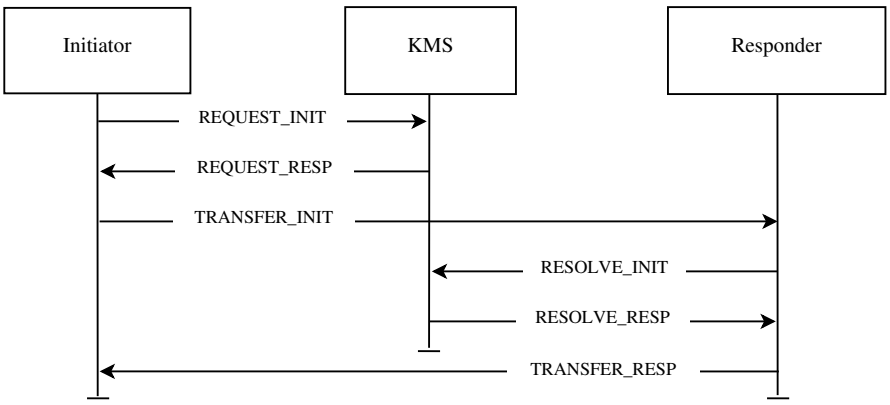


Figure 2.6: MIKEY-TICKET in Full Three Round-Trips Mode

Since MIKEY-TICKET is based on use of a trusted KMS, it is well suited to serving large numbers of users. Moreover, it is also possible to externalize the KMS and hence the basis of trust can be located outside of IMS, i.e. the trust in KMS can be independent of the trust in the IMS operator. When used in Full three round-trips and Otway-Rees like modes of operation, MIKEY-TICKET does late binding of the keys to the user and hence forking is secure, if present. It also supports deferred delivery and reuse of tickets (during the ticket's validity time period, several media sessions can be protected using the same ticket).

Table 2.3: Modes of MIKEY-TICKET

Mode	Keys shared between	Ticket generated by	Round trips	Supports Forking
1. Full three round-trips	No one	KMS	Ticket Request, Ticket Transfer, Ticket Resolve	Yes
2. Kerberos like	Responder and KMS	KMS	Ticket Request, Ticket Transfer	No
3. Otway-Rees like	Initiator and KMS	Initiator	Ticket Transfer, Ticket Resolve	Yes
4. PSK like	Initiator and Responder	Initiator	Ticket Transfer	No

2.8 SDES

The SDP Security Description for Media Streams (SDES) specifies a mechanism to signal and negotiate the cryptographic parameters for media streams in general and for SRTP in particular. It introduces a new SDP attribute called "`crypto`" which can be used by SRTP to establish cryptographic parameters in a single round-trip. Since the keys are carried within the SDP message, SDES is suitable **only** if the SDP is protected, e.g. with IPsec, TLS, SIP S/MIME [19], or similar means. Otherwise the media stream cannot be considered as secured if the keys themselves are not protected. SDES is standardized by IETF and specified in RFC 4568 [25]. 3GPP has standardized SDES as one of the key management solutions (another is MIKEY-TICKET) for media protection (see section 3.13).

The "`crypto`" attribute has the following format which describes the cryptographic suite, key parameters, and session parameters for the preceding media line.

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

e.g. a=crypto:1 AES_CM_128_HMAC_SHA1_80
       inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

The `tag` field (in this example: 1) is a decimal numeric identifier which is used to determine which of several offered crypto attributes has been negotiated.

The `crypto-suite` field (`AES_CM_128_HMAC_SHA1_80`) is an identifier that signifies the encryption and authentication algorithms.

The `key-params` field provides one or more sets of keying material. It is formatted as following

```
key-params = <key-method> ":" <key-info>
```

The `key-method` field (inline) indicates that the actual keying material is provided in the `key-info` field itself. The key-info can be expressed as following

```
<key||salt> ["|" lifetime] ["|" MKI ":" length]
```

In the above example, the first field (`PS1uQCVeeCFCanVmcjkpPywjN-WhcYD0m XXtxaVBR`) is the master key with the master salt appended and encoded in base64. The second field (2^20) indicates the lifetime of the master key and the third field (1:4) is the value of MKI and its byte length.

The session-params field is defined as a general character string and its usage is specific to any given transport. This field is ommitted in the above example.

An example of SDES key setup from [25]:

```
Offerer sends:
      v=0
      o=sam 2890844526 2890842807 IN IP4 10.47.16.5
      s=SRTP Discussion
      i=A discussion of Secure RTP
      u=http://www.example.com/seminars/srtp.pdf
      e=marge@example.com (Marge Simpson)
      c=IN IP4 168.2.17.12
      t=2873397496 2873404696
      m=audio 49170 RTP/SAVP 0
      a=crypto:1 AES_CM_128_HMAC_SHA1_80
       inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
       FEC_ORDER=FEC_SRTP
      a=crypto:2 F8_128_HMAC_SHA1_80
       inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
       inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
       FEC_ORDER=FEC_SRTP

Answerer replies:
      v=0
      o=jill 25690844 8070842634 IN IP4 10.47.16.5
      s=SRTP Discussion
      i=A discussion of Secure RTP
      u=http://www.example.com/seminars/srtp.pdf
      e=homer@example.com (Homer Simpson)
      c=IN IP4 168.2.17.11
      t=2873397526 2873405696
      m=audio 32640 RTP/SAVP 0
      a=crypto:1 AES_CM_128_HMAC_SHA1_80
       inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

In this example, the session would use the `AES_CM_128_HMAC_SHA1_80` crypto suite. However, it should be noted that SDES does not provide end-to-end media encryption when there are proxies involved as these proxies will have access to the SDES information. One method of addressing this is to use S/MIME to encode the SDES information so that only the end node can decode it.

2.9 DTLS-SRTP

Datagram Transport Layer Security (DTLS) is a channel security protocol for UDP (defined in RFC 4347 [29]) and SRTP is a security profile for RTP (see section 2.5). While SRTP has been specifically tuned for securing RTP payloads, DTLS is generic protocol. Therefore, DTLS is not as optimized for RTP as SRTP is. DTLS-SRTP (defined in RFC 5764 [30]), on the other hand, is a DTLS extension to establish keys for SRTP. DTLS-SRTP uses SRTP for encrypting RTP payloads, and DTLS for key management. RFC 5763 [31] describes a framework to use SIP to establish a SRTP context using DTLS protocol.

As shown in figure 2.7, when using DTLS-SRTP, a fingerprint attribute is transported in the SDP (e.g. as "`a=fingerprint: SHA-1 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB`") that identifies the certificate that will be presented during the DTLS handshake. In order to protect the integrity of fingerprint from modification by proxies, the SIP Identity mechanism [32] can be used. The agreement on which side acts as a DTLS client and which side acts as a DTLS server is established via SDP. The key exchange happens on the media path, independent of the signaling path, and the resulting keying material is fed into the SRTP stack.

When a certificate is presented in the DTLS handshake, then each peer can verify if the certificate matches the one used in the signaling or not. Therefore, the use of fingerprint binds the DTLS key exchange in the media plane to the signaling plane. However, this requires some form of integrity protection on the signaling plane.

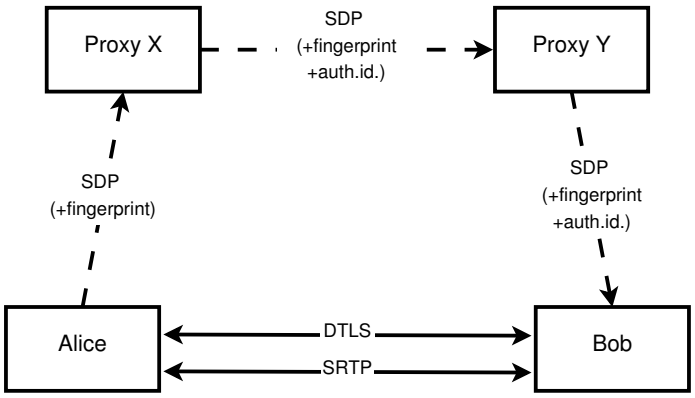


Figure 2.7: DTLS Message Exchange in SIP Trapezoid

2.10 ZRTP

ZRTP is an end-to-end media path keying protocol intended to negotiate the encryption keys to be used in a VoIP session. It is called media path keying because it is independent of support (or lack thereof) in the signaling layer and all the key negotiations occur through the RTP stream. It uses Diffie-Hellman (DH) for key exchange and the SRTP profile for encryption. ZRTP is described

in the RFC 6189 [26].

An example of ZRTP's call flow is illustrated in figure 2.8. Even though the signaling protocol does not participate in the key exchange mechanism of ZRTP, it can announce ZRTP capability via SDP attribute "*a=zrtp-hash*". If such an announcement is not present, then ZRTP attempts to perform opportunistic encryption and sends ZRTP Hello messages via the RTP session. The purpose of the Hello message is to check if the other endpoint supports the protocol and to discover a common algorithm. Any of the participants can initiated the Hello message. If a Hello response is received, then the remaining processing is carried out, otherwise it is assumed that the other party does not support ZRTP and no additional ZRTP messages are exchanged.

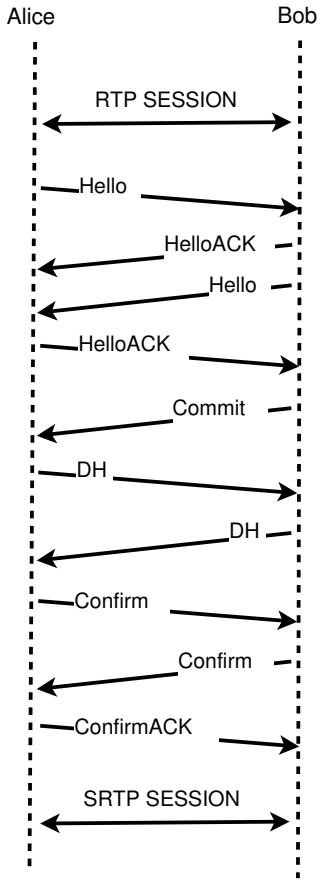


Figure 2.8: ZRTP Call Flow Example

In ZRTP ephemeral DH keys are generated for each session and therefore it does not require any Public Key Infrastructure (PKI). To protect the keys from a Man in the Middle (MitM) attack, ZRTP uses a Short Authentication String (SAS) which is generally displayed to the user and verified verbally.

The ZRTP packet format is illustrated in figure 2.9. Since its packet format is syntactically distinguishable from an RTP packet (as was shown in figure 2.2 on page 11), the concept of using Hello messages works fine. If the receiver tries

to decode this packet as a RTP packet, the version field (V) will be 0, the Padding field (P) will be 0, and the extension field (X) will be 1. Thus the receiver should check if the timestamp field is equal to the ZRTP magic cooking, i.e., the ASCII string 'ZRTP', if so, then the packet should be decoded as a ZRTP message, otherwise the packet should be discarded.

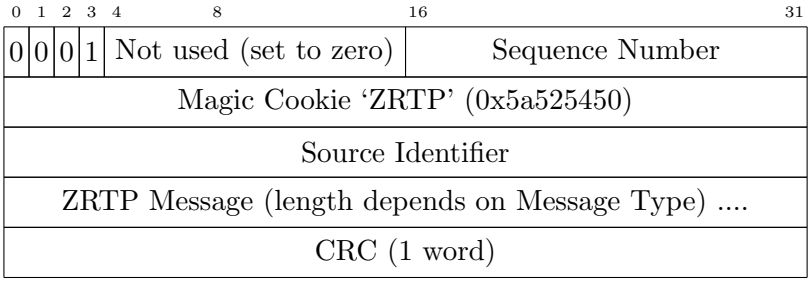


Figure 2.9: ZRTP Packet Format

The fields have following meanings:

Sequence Number is a count of ZRTP packet sent. It is used to estimate packet loss and detect packet order.

Magic Cookie is a 32 bit string that uniquely identifies a ZRTP packet. It has the value 0x5a525450 (meaning ZRTP).

Source Identifier is the SSRC number of the RTP stream that this ZRTP packet relates to.

ZRTP Message is variable length message, e.g. Hello, Commit, and so on.

CRC is a 32 bit word used to detect transmission errors.

2.11 IMS

IP Multimedia Subsystem (IMS) is a generic architecture for offering multimedia services using SIP and (S)RTP. IMS is access agnostic and therefore is a key technology for network consolidation. The IMS specification is defined in 3GPP TS 23.228 [33].

In an IMS domain, each user is assigned exactly one IP Multimedia Private Identity (IMPI) and one or more IP Multimedia Public Identity (IMPU). The IMPI, e.g. nakarmi@kth.se, is used for authentication, accounting, and so on. The IMPU, e.g. `sip:nakarmi@kth.se`, `tel:+46-6-66666666`, on the other hand is used in communications with other users.

IMS is not a service in itself, but rather it is a service enabler and it integrates different services (e.g. multimedia, instant messaging, presence, etc.) utilizing a single technology (SIP). Since IMS supports both native IP based services as well as voice services on IP, it offers network operators a significant reduction in the cost of operating their networks. Moreover, since IMS is uses the same protocols as the Internet, it is easier for developers to create applications using the already existing Internet protocols. Explaining all the details of IMS is not

possible in a short section, for more details refer to 3GPP's IMS specification [33]. Ericsson's white paper [34] outlines the value of using IMS.

2.12 GBA

3GPP's Generic Bootstrapping Architecture (GBA) is a mechanism that provides bootstrapping of application security for subscriber authentication. It is based on the Authentication and Key Agreement (AKA) [35] protocol. The GBA specification is 3GPP TS 33.220 [36]. The specification mentions two types of GBA: GBA_ME and GBA_U. In the case of GBA_ME, the UICC in the mobile device is GBA unaware and all GBA specific functions are carried out in the Mobile Equipment (ME), whereas in the case of GBA_U, the GBA specific functions are split between the ME and UICC.

Figure 2.10 shows the network elements involved in the bootstrapping architecture. It is important to notice the difference between GBA and Generic Authentication Architecture (GAA). The GAA enables the Network Application Function (NAF) to re-use the bootstrapped authentication and to agree on a shared secret with the User Equipment (UE).

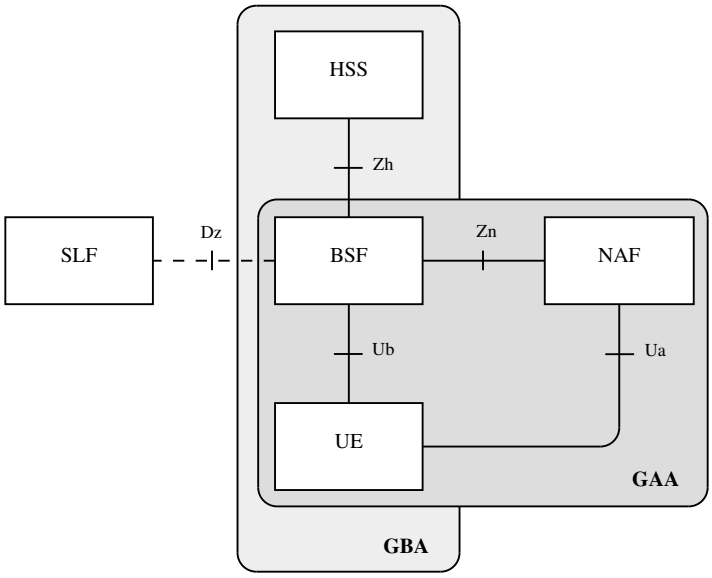


Figure 2.10: Network Elements for Bootstrapping with GBA and GAA

A brief description of these network elements:

User Equipment (UE)

The UE is the terminal equipment that participates in bootstrapping process (figure 2.11) using the UICC to establishe a temporary shared secret Ks with Bootstrapping Server Function (BSF). It also generates a key, Ks_NAF, to authenticate messages exchanged with the Network Application Function (NAF).

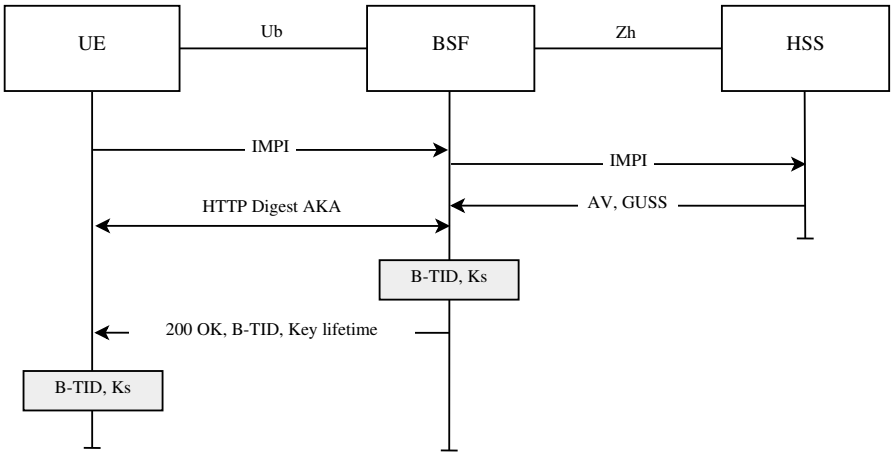


Figure 2.11: Bootstrapping Process

Bootstrapping Server Function (BSF)

The BSF participates in the bootstrapping process (figure 2.11) with the UE to establish a temporary shared secret Ks. It facilitates the bootstrapping usage process (figure 2.12) by supplying the NAF with an appropriate key (Ks_NAF) to authenticate the UE. It also acquires the GBA User Security Settings (GUSS) from the HSS.

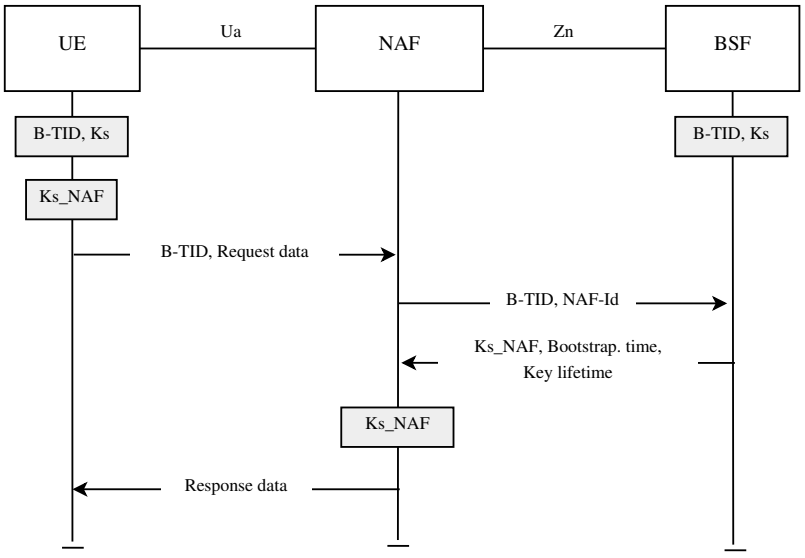


Figure 2.12: Bootstrapping Usage Process

Network Application Function (NAF)

The NAF supports GBA based user authentication and participates in bootstrapping usage process (figure 2.12). It communicates securely with the BSF to acquire the key (Ks_NAF).

Home Subscriber System (HSS)

The HSS stores information related to each subscriber. It performs the combined job of Home Location Register (HLR) + Authentication Centre (AuC) in Global System for Mobile Communications (GSM). Communication with the HSS uses the Diameter protocol [37].

Subscriber Locator Function (SLF)

The SLF provides the BSF with the name of the HSS that is to be used, if there is more than one HSS in the domain. The BSF may be configured to use a pre-defined HSS

The interfaces between the network elements are described below:

Ua The Ua interface between UE and NAF is used for for GAA. This interface supplies Bootstrapping Transaction Identifier (B-TID) to the NAF. It uses an application-specific protocol secured by Ks_NAF. These protocols include: HTTP digest authentication [38], HTTPS [39], and PKI [40].

Ub The Ub interface between UE and BSF is used for GBA. It establishes a security association between the UE and the BSF, i.e. it establishes B-TID and Ks. This interface uses HTTP Digest AKA and is described in 3GPP TS 24.109 [38].

Zh The Zh interface between BSF and HSS is used for GBA. It enables the BSF to retrieve an Authentication Vector (AV) and GUSS from the HSS. This interface is based on Diameter protocol and is described in 3GPP TS 29.109 [41].

Zn The Zn interface between BSF and NAF is used for GAA. It enables the NAF to retrieve key material and user security settings from the BSF. This interface is based on Diameter/Web Services and is described in 3GPP TS 29.109 [41].

Dz The Dz interface between BSF and SFL is used to retrieve the name of the HSS.

2.13 Summary

In this chapter, we discussed VoIP and the protocols related to VoIP. SIP is a signaling protocol which is used for management of multimedia sessions. Similarly, SDP is a media description protocol which is used for describing the multimedia sessions. Used together with SIP and SDP, RTP is a transport protocol for delivering real-time data over IP networks. SRTP, on the other hand, is a security profile for RTP. SRTP provides confidentiality, authentication and replay protection to RTP traffic. MIKEY is one of the key management

protocols which is used to establish encryption keys for security protocols, specifically SRTP. MIKEY-TICKET is also a key management protocol which extends MIKEY and uses a trusted KMS for ticket-based key distribution. SDES and ZRTP are yet another key exchange mechanisms. While SDES establishes cryptographic parameters via SDP, ZRTP uses RTP streams to exchange the keys. IMS is an architecture for offering multimedia services using SIP and (S)RTP. GBA is a mechanism used in IMS that provides bootstrapping of application security for subscriber authentication.

Chapter 3

Related Work

This chapter presents some theses, reports and standards that are related to the evaluation and implementation of secure VoIP. It summarizes some of the related theses and reports. Then it presents specifications by IETF and 3GPP. Finally it mentions some of the application and libraries that are relevant to this thesis project.

3.1 Initial SRTP Performance Measurements

In his master's thesis [42], Israel Abad Caballero discussed and evaluated a security model for Mobile VoIP and addressed both signaling protocol (SIP) as well as the data transport protocol (RTP). The evaluation presented in his thesis focused on SRTP and its effects on media processing. The tests were conducted using a 700 MHz Pentium III processor machine with 112 MB RAM and with his SRTP implementation (called MINIsrtp) was integrated into the minisip [43] SIP user agent (UA).

He argues that the additional 4 bytes that SRTP transmits (if the authentication tag is present) adds negligible time for transmission of the SRTP packets as compared to RTP packets. His measurements show that packet creation time for RTP+SRTP took ~80 μs as compared to ~5 μs for ordinary RTP packet, hence the difference in processing time per packet is small. Therefore, he concludes that the ultimate impact on performance and transmission is imperceptible.

He also makes several suggestions to improve VoIP security. These suggestions are:

1. Improve session security by using:
   - DNSSEC to secure DNS look-ups
   - TLS to protect SIP transactions
2. Improve media security by using:
   - MIKEY as the key-management protocol
   - SRTP+AES to protect media stream

3.2 Initial MIKEY Performance Measurements

In his master's thesis [44], Johan Bilien discussed and measured the additional delay required for key exchange during call establishment when using MIKEY (Note that at the time MIKEY was in the process of being standardized). He also discussed how MIKEY can be used when there is session mobility and/or device mobility.

His measurements after adding security features were presented in a separate co-authored paper [45] (these results are discussed in section 3.5).

3.3 SRTP and ZRTP Performance Measurements

Alexander, Wijesinha, and Karne have presented their experiments on the performance of SRTP in [46]. Their experiments were conducted using Windows-based snom[1], Linux-based Twinkle [48], and bare PC softphones [49]. The snom and bare PC softphones used SDES/SIP for key exchange; whereas Twinkle used ZRTP.

The processing times were measured on the bare PC softphone with 128-bit AES keys and 32-bit HMAC/SHA-1 authentication tag, as well as 192 and 256-bit AES keys and an 80-bit authentication tag. The VoIP performance was evaluated on the snom, Twinkle, and bare PC softphones with 128-bit AES key and a 32-bit authentication tag. The results show that SRTP processing adds less than 1 ms to RTP processing (indicating a negligible increase in processing time due to SRTP) and the throughput is 81.6 kbps without SRTP, and 83.23 kbps with SRTP (indicating no significant alteration in throughput). Note that the increased data rate for the case of SRTP is due strictly to the additional authentication field which is included with each RTP packet.

They concluded that the authentication processing is more expensive than encryption regardless of key/tag-size and that the addition of SRTP protection to VoIP traffic over RTP has a negligible effect on voice quality, in terms of either jitter or packet inter-arrival time.

3.4 Security Analysis of MIKEY-TICKET

Oscar Olsson, in his master's thesis [50], has done a security analysis of MIKEY-TICKET and offered some recommendations. Oscar performed his analysis by focusing on the symmetric-key variant and the two-party case of MIKEY-TICKET. MIKEY-TICKET was still an IETF draft version during his thesis work. Oscar concluded that the protocol is secure in the realistic setting of multiple sessions running in parallel in an adversary controlled network.

[1] A reference for snom is missing from the original paper. We found [47] during our Internet search for snom.

3.5 Call Establishment Delay for Secure VoIP

In [45], Johan Bilien, Erik Eliasson, and Jon-Olov Vatn presented the effect of MIKEY authentication handshake and SRTP session key generation on the call setup delay. They have presented the measurements of the call setup delay for their own implementation of MIKEY and SRTP protocols. The test-bed UAs were running on 1.4 GHz Pentium IV machines and measurements were taken in terms of the MIKEY Response in the 200 OK message.

The measurements show that the calling delay is increased by about 4 ms and the answering delay is increased by around 10 ms. Thus the authors suggest that call setup delay will not be significantly affected by introducing these security protocols.

3.6 A Secure VoIP User Agent on PDAs

In [51], Bilien, Eliasson, and Vatn give an overview of secure VoIP measurements from their earlier papers ([42] and [45]) of delays related to call setup delay and media protection. Their measurements were based on UAs running in PCs.

With regard to their experience running the same UA on a PDA the authors presented only a general discussion regarding battery power consumption and audio quality, but have not given measurements of the performance of their implementation when running on a PDA. The PDA was a HP iPAQ h5550 running Familiar Linux[2] and minisip was the SIP UA.

3.7 Secure VoIP: Call Establishment and Media Protection

In [52], Bilien, Eliasson, Orrblad, and Vatn discuss different security services relevant for VoIP and have presented their measurements of secure call establishment for MIKEY, SRTP, and IPsec. Their measurements are based on a minisip [43] UA running on a 500 MHz Pentium III machine. They conclude that the call establishment delay will not be significantly affected by introducing these security protocols.

In their implementation for a keying protocol, the MIKEY messages are carried as a multi-part MIME body in the SIP message and not as an SDP attribute. They discovered that when TLS is used, digital signatures of MIKEY messages and their verification take less time than when TLS is not used. They attribute this difference in delay to the pre-initialized crypto library and cached certificates/keys (as this shifted some of the computation to the TLS tunnel's initiation - thus removing this delay from the MIKEY computations).

Their measurements show that initial ringing delay was ~80 ms for both

[2] The official Familiar web page `http://familiar.handhelds.org/` is currently under maintenance as of this writing

IPsec and SRTP which is insignificant to a human user. However, the per RTP packet delays for both the caller and the callee are found to be higher when using IPsec than when using SRTP. Since these results are implementation dependent the authors do not draw the conclusion that IPsec in general leads to higher per RTP packet delays than SRTP.

Based upon the measurements of their implementation they suggest:

1. Use SRTP for media protection
2. Use S/MIME and MIKEY for end-to-end authentication and keying
3. Use TLS for hop-by-hop protection of SIP messages

The authors suggest using SRTP to protect the media streams because it is easier to write portable implementations which can be independent of the IPsec support provided by the end-host system. Additionally, the application can know if it implements SRTP and hence has end-to-end security, but the application can not easily know if the IP stack has an IPsec tunnel for the end point with which it is communicating.

3.8 Secure VoIP Performance on Handheld Devices

In [53], Erik Eliasson presents his performance measurements of a secure VoIP UA running on an HP iPAQ h5550, running the Familiar Linux distribution and connected wirelessly. He reports measurements for both call-setup and media-processing.

Minisip was used as the UA. This means that the caller's computation of the session key starts before the 200 OK response is received and the MIKEY messages are exchanged before the callee's phone starts ringing. These features minimize the media clipping and the risk of ghost ringing. Media clipping occurs when the media is not delivered at the start of the session. Ghost ringing occurs when the ringing starts, but the session is not subsequently initiated. Avoiding ghost ringing requires reliable provisional reponses. The call-setup measurements show that there is ~3 seconds of ringing delay and the caller's session key calculation takes ~1600 ms as compared to the callee's which takes ~400 ms. These delays are large and noticeable by user. The time required to compute the session keys affects the number of audio packets that are lost (leading to media clipping - since there were no keys to encrypt the audio until the session key and derived keys care available). Because the session key computation time will increase with decreasing processing power, this is potentially an issue for low performance processors. The authors suggest that code-optimization of minisip for handheld devices can improve the performance considerably. However, the authors did not do any code optimization.

Regarding the media processing, this report focused on the overhead of the security protocols as compared to the rest of the processing steps, e.g. silence detection/suppression and echo cancellation. In their measurements the processor took only ~0.07 ms to encrypt 20 ms of audio data and ~0.36 ms to do per packet authentication processing. The total processing time on both the

caller and callee side was ~1.3 ms. As a result the iPAQ hardware had no problem handling several simultaneous VoIP calls.

This work differs from ours in that we have explored various alternatives and derived recommendations to achieve VoIP security in the current generation of mobile devices (based on Android). We have also broadened the research by making the system complaint to the standards used in 3GPP's IMS (see section 1.2).

3.9 Evaluation of Secure Internet Telephony

In his licentiate thesis [54], Erik Eliasson has shown a way to implement end-to-end secure VoIP using open standards - TLS for the signaling, SRTP for media, and MIKEY to do an authenticated session key exchange. The use of IPsec to transport the media has also been implemented and evaluated.

His thesis builds upon his five of earlier works, already summarized in sections 3.5, 3.7, and 3.8. His performance measurements show that secure VoIP can be implemented both on PC hardware and devices with relatively low processing power such as the HP iPAQ PDA.

3.10 Alternatives to MIKEY/SRTP to Secure VoIP

In [55], Joachim Orrblad examines IPsec as an alternative to MIKEY/SRTP and shows how to integrate the key exchange for IPsec in the SIP call signaling. He concludes that while IPsec may be valuable for its ability to protect general traffic (not only the media streams) and even though SIP initiated IPsec makes it possible to establish IPsec tunnels between two VoIP peers, SRTP should still be used for media protection.

He favors SRTP over IPsec for following reasons:

- With IPsec, the UA is dependent upon a particular IPsec implementation and/or operating system that it is running on.
- His measurements revealed that with IPsec, there is loss of packets for ~0.7 seconds in the beginning of the call. However, this could be due to his particular implementation.
- IPsec causes problems for some NAT and firewall devices.
- IPsec offers host-to-host security, where as SRTP offers application-to-application security.

3.11 Mobile Web Browser Extensions

In [56], Tomas Joelsson has shown the use of local web server (proxy), running as a background process on a mobile phone, to add new functionality to web applications running in the phone's built-in browser. The author has implemented a MIDlet [57] to communicate with both local browser and remote server.

This work is relevant to us in the sense that it deals with the use of a local proxy in mobile device. As will be discussed in the section 4.4.4, one of the implementation strategies for our work is to use a proxy for intercepting both signaling traffic and media. Although the implementation platform, proxy functions, and deployment environment will be different from this earlier work, the concepts in his thesis are helpful.

3.12 Key Management Extensions for SDP and RTSP

The extension to SDP defined in IETF RFC 4567 [58] is relevant to this thesis because it provides a mechanism to carry messages specific to a key management protocol in SDP. Although the specification presents the use of this mechanism with MIKEY, the mechanism is applicable to MIKEY-TICKET **without** any modification (because MIKEY-TICKET is just a mode of MIKEY).

The SDP extension in its simplest form looks like following:
Format : `a=key-mgmt:<prtcl-id> <keymgmt-data>`
Example : `a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...`

The `key-mgmt` is a string indicating this is a key management attribute, while `prtcl-id` is a protocol identifier and `keymgmt-data` is a message specific to the key management protocol used and is encoded in base64 [59]. This attribute may be used at session level, media level, or at both levels.

This specification is more powerful than "`k=`" field in SDP because a key management protocol generally has more parameters than just a key, hence the "k=" field is not sufficient. Moreover, this specification is *independent* of the signaling protection used and hence it is better than SDES which requires the keying material to be protected along with the signaling.

3.13 3GPP TS 33.328 IMS Media Plane Security

3GPP TS 33.328 [4] defines a standard for IMS media plane security that provides end-to-end protection of RTP payload over any network. The specification uses SRTP as its security protocol and has two key management solutions. The first one is based on SDES and the second one is based on a Key Management Service (KMS). The KMS based solution is relevant to this thesis.

The KMS based solution uses MIKEY-TICKET as described in section 2.7. GBA-GAA (as discussed in section 2.12) acts as the reference model, where the KMS performs the role of a NAF. This solution is attractive because it is independent of the SIP signaling security and is scalable to large numbers of users. Figure 3.1 summarizes the processing involved in this solution. The interaction with KMS includes the `REQUEST_INIT`, `REQUEST_RESP`, `RESOLVE_INIT`, and `RESOLVE_RESP` messages of MIKEY_TICKET (as illustrated in figure 2.6 on page 17). These Ticket Request/Resolve key management messages are based on HTTP. Similarly, SDP offer/answer exchange includes `TRANSFER_INIT` and `TRANSFER_RESP` messages. These Ticket Transfer/Response key management messages are based on SIP using the SDP extension described

in section 3.12.

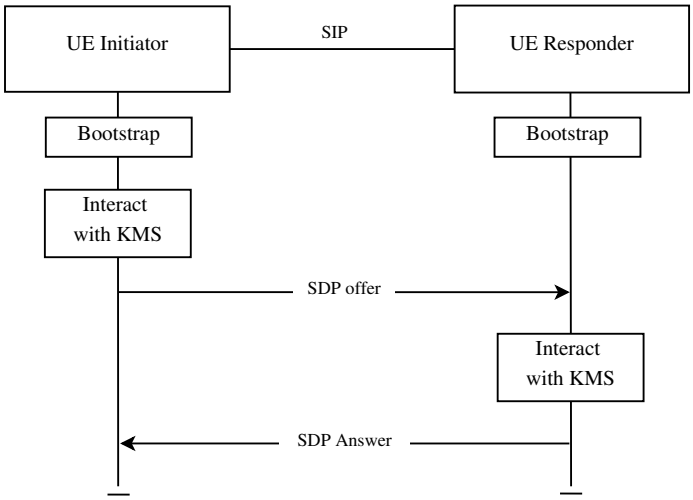


Figure 3.1: KMS Based Solution for Media Plane Security [4]

3.14 3GPP TR 33.914 using SIP Digest in IMS

The current GBA standard (see section 2.12) provides a bootstrapping mechanism that is limited to UICC-based credentials. However, the most popular authentication mechanism in most IMS deployments is SIP Digest (SD) (see RFC 3261 [19]). The use of SD credentials means that there is no need for a separate security infrastructure and even fixed devices (with no UICC) can use the bootstrapping mechanism. Therefore, in order to enable GBA bootstrapping using SIP Digest, 3GPP has started a study (3GPP TR 33.914 [60]) on an extension of GBA known as GBA Digest.

The interface between the network elements have been summarized in table 3.1. The password that is used for the SD digest response calculation is proposed to be calculated as shown in Equation (3.1). The Key Derivation Function (KDF) used in this Equation is described in 3GPP TS 33.220 [36].

$$password = KDF(H(A1), \text{“}GBA\_Digest\_RESP\text{”}) \tag{3.1}$$

Another proposal in TR 33.914 is the caculation of `Ks` as shown in Equation (3.2). In order to prevent man-in-the-middle (MITM), `TLS_MK_Extr` (to be extracted from TLS master key) has been used. For more details on this, refer the original document 3GPP TR 33.914 [60].

$$Ks = KDF(H(A1), TLS\_MK\_Extr, \text{“}GBA\_Digest\_Ks\text{”}, Digest - response) \tag{3.2}$$

As of this writing, TR 33.914 has not yet been completed. Chapter 4 and chapter 5 in this thesis describe what assumptions have been made and how GBA Digest has been implemented for authentication bootstrapping.

Table 3.1: Potential Interfaces between the Network Elements in GBA Digest

Interface between	Remarks
UE and NAF	It is application specific. Ua from GBA can be reused.
UE and BSF	Ub in GBA needs to be extended to use SD.
NAF and BSF	Zn from GBA can be reused.
BSF and HSS	BSF now needs SD-AV from HSS. There are two options to achieve this: 1. Extended Zh in GBA 2. Use Cx interface (see 3GPP TS 29.228 [61] and 3GPP TS 29.229 [62])

3.15 Existing VoIP Applications and Libraries

Some of the VoIP applications and libraries that are relevant are shown in table 3.2. We will consider using one or more of these in our design and implementation, described in chapter 4 and chapter 5.

Table 3.2: Some Relevant VoIP Applications and Libraries

Name	Type	Encryption	Platform	Reference
Minisip	Application, libraries	SRTP, MIKEY	PC, handset[a]	[43]
Sipdroid	Application	N/A	Android	[63]
Nebula	Application	N/A	Android, Linux[b]	[64]
SIP Communicator	Application	ZRTP	PC	[65]
Twinkle	Application	ZRTP	Linux only	[48]
Zfone	Software[c]	ZRTP	PC	[66]
libSRTP	Library	SRTP	Linux	[67]
Zorg	Library	ZRTP	PC, mobile	[68]

[a] Minisip had been tested in HP iPAQ [51]
[b] Linux version of Nebula is based on Minisip
[c] Zfone works on top of existing SIP+RTP programs

3.16 Summary

In this chapter, we discussed some theses and summarized their findings. These theses showed that SRTP had very good performance both in PC and HP iPAQ, and suggested using SRTP for media protection. The results presented in some of these theses showed that the call setup delay in PC was not significantly affected by introducing security features. But the call setup delay in HP iPAQ was, however, large and noticeable by the user. Besides these, we also discussed how RFC 4567 extends standard SDP to carry key management messages. Then

we summarized 3GPP's specification (TS 33.328) for media plane security in IMS. 3GPP's TS 33.328 has standardized to use SDES and MIKEY-TICKET as key management protocols, and SRTP as security protocol. We also discussed the ongoing 3GPP's standardization effort on extension of GBA known as GBA Digest, which is supposed to use SIP Digest for bootstrapping mechanism. Finally, we mentioned some VoIP applications and libraries, one or more of which are considered in our design and implementation.

Chapter 4

Design

This chapter presents the alternatives and choice of various technologies/ components that will be used for implementation. First, it discusses about choosing various protocols for signaling, transport, security, and key exchange. Then, it discusses on authentication mechanisms. Finally, it presents the system components and describes the operational flow.

4.1 Device Platform

Regarding the choice of mobile device platform, we decided to worked on Android. The motivation for choosing Android was driven by its very rapid growth (888.8 % growth in 2010, and forecasted to be the number one mobile operating system by 2014 [69]).

4.2 Signaling Protocol

H.323 is a ITU-T recommendation for multimedia communication. It describes how a number of protocols fit together to provide a multimedia conference. H.323 uses signaling concepts from the telecommunications industry. SIP on the other hand covers only the signaling parts of H.323. SIP is an Internet-centric protocol and is simpler than H.323. For these reasons, SIP has been widely deployed for initiating, terminating, and controlling multimedia sessions over IP. SIP has also been accepted as the signaling protocol for IMS. Therefore, this thesis focuses on SIP as the signaling protocol.

4.3 Transport Protocol

RTP is one of the technical foundations of VoIP and is used together with H.323 or SIP. One of the main advantages of using RTP is that new multimedia formats can be added without changing the underlying standard. This thesis focuses on RTP as the media transport protocol.

4.4 Security Protocol

SRTP is a well established security protocol for RTP based traffic. 3GPP has standardized on using SRTP in IMS irrespective of the key management solution. Therefore, this thesis adopts SRTP as the media security protocol.

A VoIP application produces RTP data in the application layer (see figure 4.1). The data moves down to the transport layer before finally leaving the device via the network access layer (the link and physical layer are hidden by the operating system). We, therefore, have opportunities to work on different layers, each of which presents various alternatives for converting RTP into SRTP. The following sections present some ideas concerning alternatives for how to implement this RTP to SRTP conversion at each of these different layers.

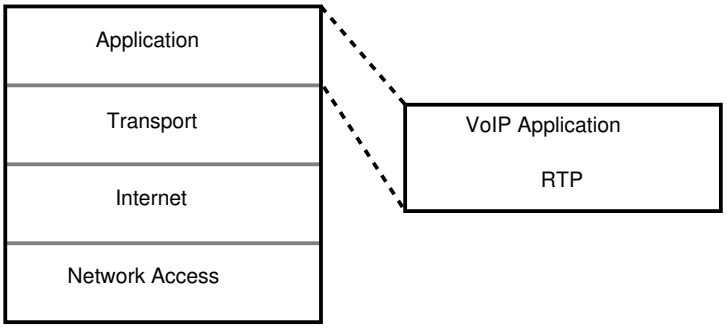


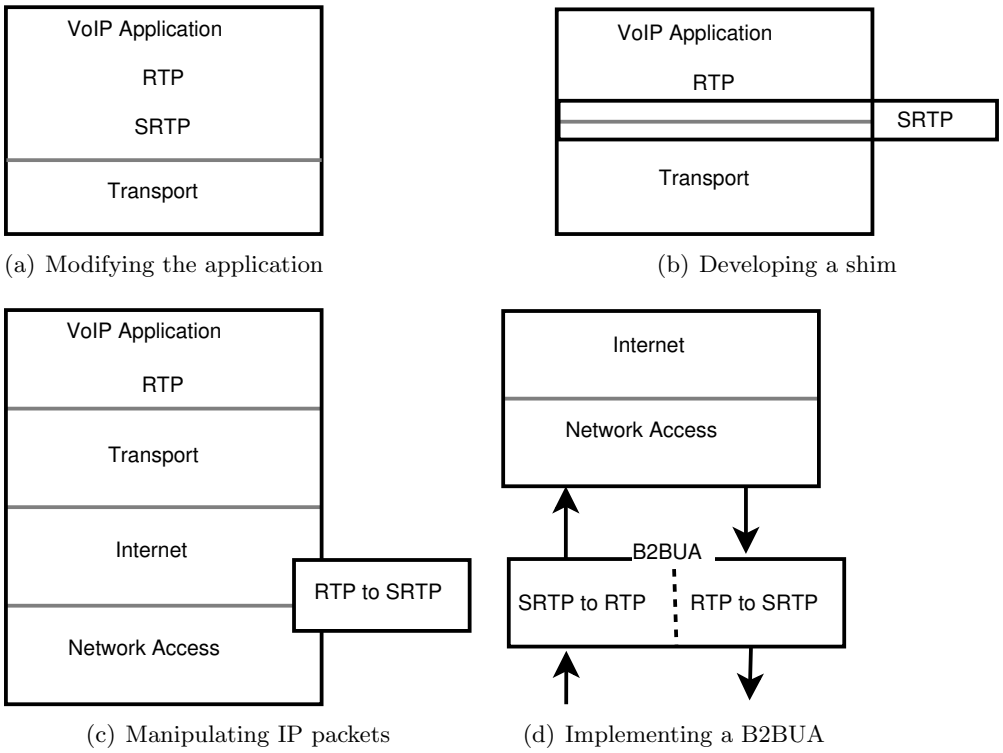
Figure 4.1: VoIP Application in TCP/IP Layer

4.4.1 Strategy 1 - Modifying the Application

In this approach we integrate SRTP into the VoIP application (see figure 4.2(a)). This approach involves modification of the application so that the RTP packets are transformed into the corresponding SRTP packets *within* the application. If the source code of the application is available, then this approach should be the most efficient.

4.4.2 Strategy 2 - Developing a Shim

Another alternative is to intercept library calls in order to change the behavior of an application. The library or application which does this is known as a shim (see figure 4.2(b)). In this approach, a suitable shim will intercept the system calls from the VoIP application so that when it reads/writes the socket, the necessary conversions are done between RTP and SRTP packets. As a result the SRTP packets are generated in the application layer before being passed to the transport layer. This approach has been used frequently in the past to realize various solutions, such as SOCKS [70]. To my knowledge, Android phones need to be rooted to implement this strategy.



(a) Modifying the application

(b) Developing a shim

(c) Manipulating IP packets

(d) Implementing a B2BUA

Figure 4.2: Alternative Approaches for Media Protection in Handset

4.4.3 Strategy 3 - Manipulating IP Packets

Another approach is to capture and modify the IP packets, in a similar fashion to how a firewall captures the IP packets at the Internet layer and drops or modifies them. The idea would be to replace each RTP packet with a corresponding SRTP packet (see figure 4.2(c)). The success of this approach assumes the ability to intercept and manipulate IP packets. To my knowledge, Android phones need to be rooted to implement this strategy [71].

Since this approach processes RTP data present in each IP packet, there may be problems during datagram fragmentation, i.e. when a single RTP datagram is spread over more than one IP packet. Nevertheless, generally for audio data, say for packetization times of 20–100 ms @ 8000 Hz, the RTP payload ranges in size from 160-800 bytes which is much less than a typical MTU (1500 for Ethernet [72], ~2200 for IEEE 802.11 [73], and upto 1500 for 3G [74]). As such, there should not be any problem due to datagram fragmentation in most networks.

4.4.4 Strategy 4 - Implementing a B2BUA

The idea is to implement a B2BUA that acts as intermediary (proxy) for both the SIP signaling and the media transfer (see figure 4.2(d)). The B2BUA resides in the mobile phone and runs as a local server. The VoIP application is configured to connect to the B2BUA (e.g. 127.0.0.1:5060) and the B2BUA in

turns connects to the SIP server (e.g. sip.server:5060). The VoIP application, therefore, sees the B2BUA as a SIP proxy server *as well as the remote peer*. To achieve this the B2BUA manipulates the SIP messages to redirect all the SIP traffic via itself. Similarly, after the call establishment, the B2BUA does the RTP to SRTP and SRTP to RTP conversion of the media between the remote peer and the local VoIP application.

Since the strategy of modifying the application does not require rooting the Android phone and is simpler to implement than other strategies mentioned here, this thesis focuses on modifying the open source VoIP client Sipdroid [63]. The development of Sipdroid is very active. Moreover, the author already has familiarity with the codes of Sipdroid while working on his previous project [64]. Hence the modification can be done in less time, unlike if other VoIP clients were chosen (Note that IMSDroid [75] is an open source 3GPP IMS Client for Android devices, and would also have been a good choice).

4.5 Key Exchange Protocol

One of the potential alternatives for a key exchange protocol is ZRTP (see section 2.10). ZRTP is interesting because it is multiplexed on the same port as RTP and is indifferent to the support for media security in the signaling protocol. However, ZRTP has some practical issues. First, ZRTP uses a Short Authentication String (SAS) (which is generally displayed to the user and verified verbally). To verify SAS with everyone is unfeasible in a group communication. Next, ZRTP needs an established RTP session for key exchange. But IMS does not allow media to be sent through the network before the signaling is done. When using ZRTP, the key exchange can suffer from this delay. ZRTP is also naturally prone to issues like ghost-ringing and media-clipping (see report [53]).

Another alternative is DTLS-SRTP (see section 2.9), but it is dependent on the integrity protection of signaling plane. Like ZRTP, DTLS-SRTP also performs the key exchange in the media stream. Hence, it also suffers from the delay before the RTP media can flow. Similarly, both ZRTP and DTLS-SRTP do no favor lawful interception (LI). Both of them are also **not** suitable for group keys (since only individual peer-to-peer keys are provided). Note that DTLS-SRTP has been already **not** chosen for IMS, even though it was considered in a technical report 3GPP TR 33.828 [76].

On the other hand, 3GPP has standardized SDES (see section 2.8) and MIKEY-TICKET (see section 2.7) as key management solutions for media protection. Note that with SDES and MIKEY-TICKET, it is possible to perform lawful interception (LI). While SDES relies on the SIP signaling security, MIKEY-TICKET is independent of the signaling and transport network. Moreover, MIKEY-TICKET is based on use of a trusted KMS, and scales well to serve a large numbers of users. Therefore, MIKEY-TICKET is more appropriate for end-to-end security.

This thesis focuses on using MIKEY-TICKET as the key exchange protocol.

4.6 Authentication Mechanism

This thesis takes into account both GBA (UICC based, see section 2.12) and GBA Digest (SIP digest, see section 3.14). For experiments with GBA, a vitual UICC was used. For GBA Digest, the necessary modification to the network components and interfaces were made to use SD. In the case of BSF-HSS communication, `Zh` was extended for SD instead of using `Cx`. The reason to choose `Zh` is that it allows operators to support GUSS; whereas the usage of GUSS is not possible with `Cx`.

4.7 System Components

The components of the system are illustrated in figure 4.3. The Mobile Handset is a UE which is used to make VoIP calls. The HSS and the BSF participate in bootstrapping (GBA and GBA Digest) whereas the KMS (acting as NAF in GAA) issues and resolves Tickets for MIKEY-TICKET based key exchange.

The UE provides a User Interface (UI) for making and handling calls. It also contains a SIP stack which takes care of SIP signaling. Similarly, recording and playback of voice is done by a RTP/CODECs stack. Conversion of RTP to SRTP and SRTP to RTP is handled by SRTP Profiler.

The KMS and UE both contain a MIKEY-TICKET Enabler and a GBA Enabler in order to use MIKEY-TICKET messages and GBA/GAA respectively. Section 5.2 presents the details of each component.

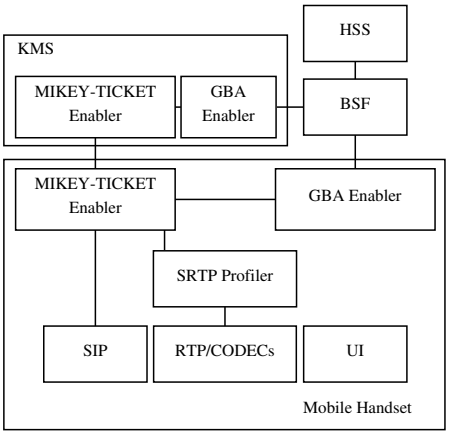


Figure 4.3: System Components Diagram

4.8 Operational Flow

Figure 4.4 illustrates the operational flow of messages in the system for both the Initiator (notation to right and on top of line) and the Responder (notation to left and at the bottom of line, enclosed in curly braces).

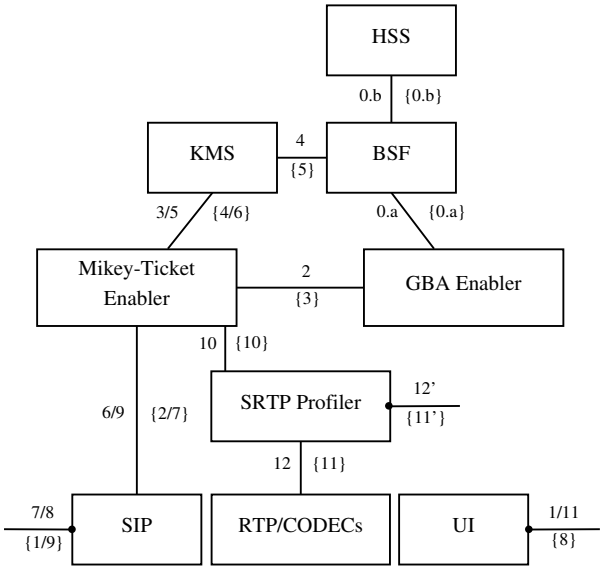


Figure 4.4: Operational Flow

Initiator side (Full three round-trips with Forking enabled)

0. a,b Bootstrapping process. If the existing B-TID is still valid, it can be skipped (that is why this is labeled as the zeroth step).
1. User initiates the call
2. MIKEY-TICKET API gets B-TID and Ks_NAF from GBA Enabler
3. TICKET_INIT message, containing the B-TID (Note that if the existing Ticket is valid, the same Ticket can be used)
4. KMS gets Ks_NAF for Initiator from BSF
5. TICKET_RESP message, containing the Ticket
6. Ticket is embedded in SDP payload of INVITE message, i.e. TRANSFER_INIT message and security is indicated by SAVP in media line attribute
7. INVITE message is sent
8. Response message is received, if it is not 200 OK, call does not succeed
9. TRANSFER_RESP message is extracted from SDP
10. If TRANSFER_RESP message can be parsed and validated, session keys are available for SRTP, otherwise call does not succeed
11. User is notified
12. 12, 12' incoming SRTP packets are decoded to RTP and outgoing RTP packets are encoded to SRTP

Responder side (Full three round-trips with Forking enabled)

0. a,b Bootstrapping process. If the existing B-TID is still valid, it can be skipped (that is why this is labeled as the zeroth step).

1. INVITE message is received, with TICKET_INIT message
2. SAVP is detected and ticket in the SDP is extracted
3. MIKEY-TICKET API gets B-TID and Ks_NAF from GBA Enabler
4. RESOLVE_INIT message
5. KMS gets Ks_NAF for Responder from BSF
6. RESOLVE_RESP message
7. If step no. 6 does not succeed, then the call does not succeed neither, so CANCEL is sent, otherwise 200 OK is prepared to be sent
8. User is notified of call, if he denies, call does not succeed.
9. TRANSFER_RESP message
10. Session keys are available for SRTP
11. 11, 11' incoming SRTP packets are decoded to RTP and outgoing RTP packets are encoded to SRTP

4.9 Summary

In this chapter, we presented the design of system. The final choice of technologies has been motivated by industry standards and fulfillment of the purpose. SIP (RFC 3261) was chosen as a signaling protocol and RTP (RFC 3550) was chosen as media transport protocol. Similarly, SRTP (RFC 3711) was chosen as security protocol and Sipdroid was chosen to be modified in order to integrate SRTP in the application layer. MIKEY-TICKET (RFC 6043) was chosen as a key exchange protocol. Regarding the authentication mechanism, both the standard GBA (TS 33.220) and GBA Digest (TR 33.914) were chosen to be considered. Finally, we presented the system components and illustrated the operation flow of messages.

Chapter 5

Implementation

This chapter describes the implementation of the system according to the design as discussed in chapter 4. It presents the methodology of the implementation and describes how each components were developed and/or adapted.

5.1 Methodology

We followed the Evolutionary Prototyping Model [77] for our implementation. The choice of this model is driven by the fact that it enabled us to gradually enhanced the software with new features and improvements, while allowing me to take a functional snapshot of the software for each prototype release. This facilitated our performance evaluation for each separate component.

In order to avoid reinventing the wheel, the following artifacts were used and/or adapted:

1. Sipdroid [63] was selected as the basis of the VoIP application, it provides the SIP/RTP stack
2. libSRTP [67] was selected to implement RTP to SRTP and SRTP to RTP transformations, in order to implement SRTP
3. Ericsson's internal MIKEY-TICKET API, standalone KMS, and BSF
4. Mobile Web Security Bootstrap (MWSB) [78] for GBA API
5. Fraunhofer's FOKUS HSS [79]

The following steps were defined as checkpoints to produce distinct prototypes during the implementation of the complete system.

1. Integration of SRTP with hard-coded master key in Sipdroid
2. Integration of key management messages in SDP in Sipdroid
3. Integration of MIKEY-TICKET client in Sipdroid
4. Adaptation of standalone KMS to use hard-coded PSK
5. Integration of GBA client in Sipdroid

6. Development of Java EE KMS to perform GAA
7. Adaptation of Sipdroid, BSF, and HSS to use GBA Digest

5.2 System Components Details

Table 5.1 describes how each system component was developed or adapted.

Table 5.1: System Components Description

Components	Based on	Modification/Development
UI (Android)	Sipdroid	New settings were added to set security releted information such as KMS/BSF address. The home screen tells the user if security is enabled or not.
RTP/Codecs (Android)	Sipdroid	No changes were made to Sipdroid. A PCMA 8 kHz sampling rate CODEC was used during the measurements.
SIP (Android)	Sipdroid	When security is enabled, key management messages are included in SDP according to RFC 4567 [80]. The initiator includes a MIKEY-TICKET ticket in its INVITE message. The responder resolves the ticket when it receives this INVITE, then sends its response in the 200 OK message. SDP was removed from the 180 Ringing message.
SRTP Profiler (Android)	libSRTP	The libSRTP library is written in C. It was compiled for Android using the Android NDK and integrated with Sipdroid using Java Native Interface (JNI). The SRTP Engine was developed to interface via JNI with the library. When security is enabled RTP packets are encoded into SRTP packets before they are transferred via a UDP socket. Similarly, SRTP packets are decoded into RTP packets before they are given to the application for playback.
MIKEY-TICKET Enabler (Android)	MIKEY-TICKET API (Ericsson)	The MIKEY-TICKET API is written in C. It was compiled for Android using the Android NDK and integrated with Sipdroid using JNI. MIKEY-TICKET Engine was developed to interface via the JNI with this C code. The required dependencies of this library are OpenSSL and Curl.

Continued on next page

Table 5.1 – *continued from previous page*

Components	Based on	Modification/Development
	OpenSSL	The OpenSSL distribution from [81] was used. It was compiled using Android NDK and embedded into Sipdroid as a static library.
	Curl	The original Curl distribution from [82] was used. It was compiled together with the source code of Android Froyo and used in Sipdroid as a shared library.
GBA Enabler (Android)	MWSB (Ericsson Lab)	The GBA Client API version 2 from [78] was customized and integrated into Sipdroid. See section 5.3 for more details.
KMS (Server)	MIKEY-TICKET API (Ericsson)	The KMS works as NAF in GAA. The KMS Server was deployed in the Glassfish [83] Java EE server. Ticket Request and Ticket Resolve requests are handled by a Servlet. The MIKEY-TICKET API was integrated into the server using JNI and a Glassfish specific configuration for JNI.
	MWSB (Ericsson Lab)	The GBA NAF API version 2 from [78] was used.
BSF	Ericsson Lab	The version of BSF that is deployed in Ericsson Lab was used without modification for GBA. This BSF was extended in order to support GBA Digest (see section 5.4).
HSS	FHoSS (Fraunhofer)	The FOKUS Home Subscriber Server (FHoSS) [79] was used without modification as HSS for GBA. For GBA Digest, *16777228* was registered as a new Application Identifier for the extended Zh interface (EZh). Note that the identifier *16777228* was chosen by the author only for testing. The EZh interface then returns SD-AV in its Multimedia-Auth-Answser (MAA).

5.3 GBA Enabler in UE

For GBA: The original API [78] is targeted for Java SE. A conflict between the Android SDK and the dependencies on the Apache Commons Codec [84] was solved by renaming the original Apache packages, and then refactoring the GBA Client code. Additionally, the Android Patch for XML DataType [85] had to be used.

For GBA Digest: In GBA Digest, the subscriber authentication is done by using SIP Digest instead of HTTP Digest AKA. Hence a new class

`SIPDigestClient` was made for authentication. `SIPDigestClient` is compliant with current version of 3GPP TR 33.914 [60]. Note that 3GPP TR 33.914 is not yet complete. The following derivations are assumed by author:

1. `FC` that is used in KDF is `0x02`. This should be defined. Note that `FC` is a single octet that distinguishes different instances of the algorithm. The value of `FC` for original GBA is `0x01`.
2. `TLS_MK_Extr` is a hard coded string for now. The value should be extracted from the real TLS master key. Note that `TLS_MK_Extr` denotes a session key extracted for a TLS master key. The use of `TLS_MK_Extr` binds the outer authentication protocol established by the TLS tunnel with the inner authentication established through the GBA process.
3. `Ksip_digest` is calculated as `MD5(cnonce:realm:H(A1))`. This should be defined. Note that `Ksip_digest` denotes a SIP Digest response key which is used to calculate `Bres` as mentioned in point no. 4 below.
4. `Bres` is calculated in same way as Digest-response, but with different parameters. `Bres` calculation uses `TLS_MK_Extr` as entity body and `Ksip_digest` as password. This should be defined. Note that `Bres` serves as a Message Authentication Code (MAC).
5. New directive "`bres`" is added to the response authorization header. This should be defined.

5.4 Extended BSF that Supports GBA Digest

To support GBA Digest in BSF, both the Ub and Zh interfaces were extended (see section 3.14 on page 33 and section 4.6 on page 40). The extended Zh interface enables the BSF to retrieve SD-AV and GUSS from the HSS. The Diameter application identifier *16777228* was used in the Multimedia-Auth-Request (MAR) to indicate the extended Zh (EZh). Note that the identifier *16777228* was chosen by the author only for testing; an assigned value should be asked for from Internet Assigned Numbers Authority (IANA). The Diameter aplication identifier assigned to the original Zh interface application is *16777221*. EZh also parses the MAA from HSS that now contains the SD-AV.

The extended Ub interface (EUb) uses the SD-AV to present SD authentication challenge to the UE. The SD is in compliance with 3GPP TR 33.914. Note that the assumptions presented in section 5.3 applies to EUb as well. For more details on MAR, MAA, and SD-AV, see 3GPP TS 29.228 [61] and 3GPP TS 29.229 [62].

5.5 Summary

In this chapter, we presented the implementation details of the system. We followed the Evolutionary Prototyping Model during the implementation. We presented how a secure VoIP client was implemented in Android by using and/or modifying various applications and libraries. We described how the reference

implementation of GBA Digest was done. Since the GBA Digest standardization effort (3GPP's TS 33.220) has not completed yet, we had to make necessary assumptions. These assumptions are the inputs to 3GPP's TS 33.220.

Chapter 6

Measurements

This chapter presents the measurements. It describes the test environment and the measurement methodology. Then it presents various measurements related to different processes that are involved during a secure VoIP call.

6.1 Test Environment

Figure 6.1 shows the test environment under which the measurements were done. A modified Sipdroid was installed in two Nexus One (N1) mobile phones running Android 2.3.3 GRI49. Each N1 mobile phone had 1 GHz Qualcomm® QSD8250™ processor and 512 MB RAM. They were connected to a laboratory WLAN via a WEP secured 54 Mbps IEEE 802.11g network. All the SIP messages were sent using UDP. The VoIP calls were made using a PCMA 8 kHz sampling rate CODEC.

The server components were hosted in a single laptop with a 2.6 GHz Intel® Core™ 2 Duo CPU and 3 GB RAM. The operating system was 32 bit Ubuntu 10.10 (maverick) Kernel 2.6.35-27-generic. The OpenSIPS version was 1.6.4 and MySQL version was 5.1.49. All the GBA/GAA related server components (i.e., KMS, BSF, and HSS) were deployed in a single GlassFish server 3.1 (Sun Java 1.6.0-24). A MySQL database server running on this laptop hosted the databases for HSS and OpenSIPS.

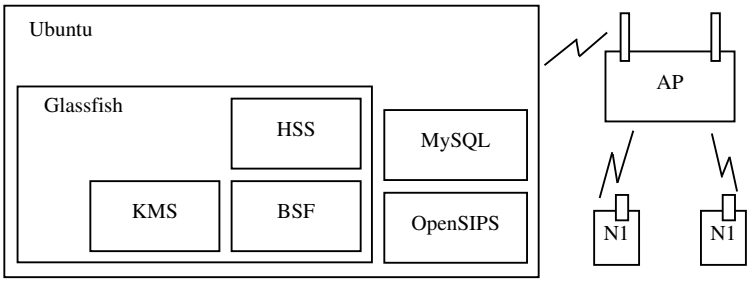


Figure 6.1: Test Environment

6.2 Measurement Methodology

Four of the measurements were related to evaluation of the delay introduced by security mechanisms. These four measurements were:

1. Measurements at the caller's side of the delay when initiating a call (see section 6.4)
2. Measurements at the receiver's side of the delay when receiving a call (see section 6.5)
3. Measurements at the caller's side of the delay when receiving 200 OK (see section 6.6)
4. Measurements of the delay for SRTP profiling of the media packets (see section 6.7)

In these measurements, original GBA bootstrapping (see section 2.12) was performed using a virtual UICC. Note that the bootstrapping was done each time and the MIKEY-TICKET was used in full three round-trips mode. Moreover, measurement of each part of the process was done along with a measurement of the total time delay. Due to these separate measurements, it is possible to estimate the time delay that would occur when bootstrapping is not done in the call sequence or when MIKEY-TICKET is used in its other modes.

For measurements `android.os.Debug` and `Traceview` were used to determine the most time consuming methods of our interest (see tutorial [86]). After this, appropriate timestamps were taken using `java.lang.System.nanoTime()`. We made a custom class `StopWatch` to make it easy to take timestamps for various methods and dump the CSV results into a file on the external memory card.

Since Android 2.3 has a built-in native SIP stack, an additional measurement was made (see section 6.8) to compare its performance with that of Sipdroid (in both insecure and secure modes). We used `tcpdump` (compiled for the ARM processor) in one of the N1 handsets to capture the packets as a call was made (note that this was the caller's handset). The `tcpdump` is a command line packet analyzer which can capture and display the network packets being transmitted or received. The packet capture file produced by `tcpdump` was then loaded into Wireshark for analysis. We calculated the ringing delay for this scenario, i.e. the time between sending the INVITE and receiving the first 180 Ringing SIP message. Note that in order to use `tcpdump`, we had to root the N1.

To compare the effect of GBA Digest, another measurement was made (see section 6.9) at both the caller's side and the receiver's side. In this measurement, we recorded only the time taken for bootstrapping because other process flows are not changed by this.

6.3 Specific Functions of Interest during the Measurements

1. `GBAEngineJava.bootStrap`

   This function is in Java. It is responsible for the bootstrapping process and includes the BSS-HSS communication. After executing this function, the UE has learned B-TID and Ks_NAF.

2. `MikeyTicketEngine.getTicket`

   This function calls the underlying function written in C via JNI. This function computes and sends the REQUEST_INIT and receives the REQUEST_RESP messages of MIKEY-TICKET. After executing this function, the initiator's UE has a Ticket that can be used in the TRANSFER_INIT message of MIKEY-TCKET.

3. `UserAgent.addKeyMgmtMessage`

   This function is in Java and is responsible for adding the MIKEY-TICKET Ticket to the SDP of the SIP INVITE message.

4. `Sipdroid.call_menu`

   This function is in Java. It comprises of all the processes for making a call, including updating the user interface. `GBAEngineJava.bootStrap`, `MikeyTicketEngine.getTicket` and `UserAgent.addKeyMgmtMessage` are called by this function.

5. `MikeyTicketEngine.getTicketResponse`

   This function calls the underlying function written in C via JNI. It computes and sends the RESOLVE_INIT and receives the RESOLVE_RESP messages of MIKEY-TICKET. After executing this function, the responder's UE has a response Ticket that can be used in the TRANSFER_RESP message of MIKEY-TICKET. The responder's UE also has the necessary SRTP master key and salt.

6. `UserAgent.onCallIncoming`

   This function is in Java. It comprises of all the processes for handling an incoming call, including updating the user interface. This functions calls `MikeyTicketEngine.getTicketResponse`.

7. `MikeyTicketEngine.verifyResponseTicket`

   This function calls the underlying function written in C via JNI. It parses the Ticket received via the TRANSFER_RESP message of the MIKEY-TICKET. After executing this function, the caller's UE has necessary SRTP master key and salt.

8. `UserAgent.onCallAccepted`

   This function is in Java. It comprises of all the processes for handling the 200 OK, including updating the user interface. This functions calls `MikeyTicketEngine.verifyResponseTicket`.

9. `SRtpEngine.protect`

   This function calls the underlying function written in C via JNI. In this experiment, each call to this function converts 160 bytes of RTP payload into SRTP.

10. `SRtpEngine.unprotect`

    This function calls the underlying function written in C via JNI. In this experiment, each call to this function converts 160 bytes of SRTP payload into RTP.

11. `Parse SDP and Read Ticket`

    These group of functions consists of all the underlying functions that parse the SDP and extract the key management message. The underlying functions are written in Java.

6.4 Measurement 1: Initiating a Call

Table 6.1 summarizes the measurements (10 measurements were made) and illustrates the effect of introducing a key management mechanism (MIKEY-TICKET) to the caller's side when initiating a call. The bootstrapping process (in Java) took longer to complete than the key management process (the latter were written in C). The high variation in total time can be attributed to the UI related updates. In the worst case, there is additional delay of ~150 ms. Given the fact that bootstrapping (~127 ms) is not done for every single call, the overhead of initiating a secure call can be less than 50 ms. Moreover, if a Ticket is reused (during its validity period) or MIKEY-TICKET is used in Otway-Rees like or PSK like modes, then the overhead can be as small as ~7 ms.

Table 6.1: Measurement Statistics at Caller's Side when Initiating a Call

Function	Secure Call	Average delay (ms)
GBAEngineJava.bootStrap	Yes	127.4
MikeyTicketEngine.getTicket	Yes	413.8
UserAgent.addKeyMgmtMessage	Yes	695.4
Sipdroid.call_menu	Yes	544.2
Sipdroid.call_menu	No	423.8

6.5 Measurement 2: Receiving a Call

Table 6.2 summarizes the measurements (10 measurements were made) and illustrates the effect of introducing a key management mechanism (MIKEY-TICKET) to the receiver's side for receiving a call. The secure call took ~80 μs more for the process of parsing SDP and reading the MIKEY-TICKET Ticket. Bootstrapping (in Java) took longer to complete than the case of caller. The additional overall delay is less than 250 ms in comparison to a non-secure call. If bootstrapping is removed from the process, then the additional delay is only

around ~80 ms. Moreover, if MIKEY-TICKET is used in Kerberos like or PSK like mode, then the delay is less than 7 ms on average.

Table 6.2: Measurements Statistics at Receiver's Side when Receiving a Call

Function	Secure Call	Average delay (ms)
GBAEngineJava.bootStrap	Yes	164.9
MikeyTicketEngine.getTicketResponse	Yes	706.3
UserAgent.addKeyMgmtMessage	Yes	626.8
Read Ticket	Yes	365.8
Parse SDP	Yes	971.9
UserAgent.onCallIncoming	Yes	624.2
Parse SDP	No	932.9
UserAgent.onCallIncoming	No	407.8

6.6 Measurement 3: Receiving a 200 OK

Table 6.3 summarizes the measurements (10 measurements were made) and illustrates the effect of introducing a key management mechanism (MIKEY-TICKET) to the caller's side with respect to the time required to receive a 200 OK. The secure call took ~70 μs more for the process of parsing SDP and reading the MIKEY-TICKET Ticket. The total process experienced an additional delay of ~4 ms. It should be noted that when TICKET_RESP is not sent (as indicated by the F bit), then there is **no** delay or additional performance cost for the caller.

Table 6.3: Measurement Statistics at Caller's Side when Receiving 200 OK

Function	Secure Call	Average delay (ms)
MikeyTicketEngine.verifyResponseTicket	Yes	325.9
Read Ticket	Yes	396.3
Parse SDP	Yes	673.7
UserAgent.onCallAccepted	Yes	107.1
Parse SDP	No	643.3
UserAgent.onCallAccepted	No	111.0

6.7 Measurement 4: SRTP Profiling

Table 6.4 depicts the performance overhead of SRTP encryption and decryption for 200 samples of data, each 160 bytes (PCMA 8 kHz). The SRTP to RTP and RTP to SRTP processing, on average, took less than 300 μs.

Table 6.4: Measurement Statistics for SRTP Profiling

Function	Average delay (μs)
SRtpEngine.protect	291.2
SRtpEngine.unprotect	281.5

6.8 Measurement 5: Ringing Delay

Table 6.5 shows the statistics of measurements (10 measurements were made) and illustrates a comparison of ringing delay while using the Android Native SIP stack and the modified Sipdroid. The measurements are based on the timestamps of packets captured by `tcpdump`. For this data we can see that the ringing delay in Android's Native SIP stack is ~250 ms less than Sipdroid for insecure calls. However, Android's Native SIP stack does not have security features. In the case of Sipdroid, the ringing delay increased by ~200 ms when security is used (note that this included bootstrapping process and MIKEY-TICKET was used in full three round trips mode).

Table 6.5: Measurements Statistics for Ringing Delay

UE	Secure Call	Average delay (ms)
Android Native	No	475.9
Sipdroid	No	708.7
Sipdroid	Yes	898.3

6.9 Measurement 6: GBA Digest Bootstrapping

Table 6.6 shows the average bootstrapping time at the caller's side and the receiver's side when using GBA Digest (10 measurements were made). These average time are similar to the bootstrapping time when using GBA (see table 6.1 and table 6.2). Therefore, using GBA Digest does not seem to introduce any additional delay in the bootstrapping mechanism. Note that `TLS_MK_Extr` was hard coded instead of extracting it from the real TLS master key. However, we think that it should not take any significant amount of time if the underlying TLS stack provides a mechanism for extracting the master key.

Table 6.6: Measurements Statistics for GBA Digest Bootstrapping

UE	Secure Call	Average delay (ms)
Caller's side	Yes	142.0
Receiver's side	Yes	165.4

6.10 Observations and Summary

The timing measurements in this chapter should be considered relative to the delays without security, rather than as absolute timing measurements. If the experiments were conducted using a 3G network and different implementations of GBA and MIKEY-TICKET, then the results would differ from the values that have been presented here. However, the results presented above are helpful to understand the relative processing time relationships between each component of the complete process of making a secure VoIP call.

Based on these measurements, we found that secure VoIP has good performance on a N1 running Android Gingerbread (2.3). In the worst case

scenario, when the secure call involves bootstrapping of both caller and receiver sides, and MIKEY-TICKET is used in full three round-trips mode, then the call setup delay was as long as ~400 ms. The time taken for the bootstrapping process could be further reduced if its implementation is done in C [1], as was done for MIKEY-TICKET. Moreover, when the previous bootstrapping is still valid for both caller and receiver sides, and MIKEY-TICKET is used in PSK like mode, then the call setup delay can be less than 20 ms. SRTP, on the other hand, had an overhead of ~300 μs for every RTP packet (containing 160 bytes of voice data corresponding to ~20 ms of PCMA 8 kHz).

The results show that less than 1 second of additional time can yield both improved security and privacy. Therefore it is a fair conclusion that the current generation of mobile devices should perform well when using the media plane security architecture specified by 3GPP (MIKEY-TICKET and SRTP).

[1] Towards the end of this thesis work, a prototype implementation of GBA Client in C was also tested in Andoid via JNI and 10 measurements were taken. The result was an average of ~40 ms for bootstrapping. This means that if all the libraries were in C, the worst-case call setup delay will be ~200 ms.

Chapter 7

Conclusions and Future Work

7.1 General

With mobile VoIP likely to have increasing adoption, mobile VoIP security will attract more attention in coming years. Today media protection has not been widely implemented in mobile consumer VoIP applications, hence this thesis is interesting both from a practical point of view (in evaluating the alternatives for media protection for mobile devices) and from a theoretical point of view (in integrating security using existing stacks and deployment).

7.2 Summary of the Work

This thesis explored the alternatives and feasibility of achieving VoIP security for mobile devices. A secure VoIP client (based on Sipdroid [63]) was produced for mobile devices running Android OS. The implementation was based on the 3GPP's IMS standards. SIP (RFC 3261 [19]) was used as signaling protocol, SRTP (RFC 3711 [1]) was used as security protocol, MIKEY-TICKET (RFC 6043 [27]) was used as key management protocol, SDP extension as defined in RFC 4567 [80] was used to carry key management messages, and 3GPP's GBA (TS 33.220 [36]) was used for authentication bootstrapping. Finally, a reference implementation of 3GPP's GBA Digest (TR 33.914 [60]) was made for authentication bootstrapping.

The prototype implementation of GBA Client written in C took only ~40 ms to bootstrap; while the Java version took ~130–160 ms for the operation. Therefore in Android, it is better performance-wise if the library code is written in C and interfaced to the application via JNI. Doing this also gives the opportunity to compile the library for other mobile platforms as well (e.g. Microsoft's Windows Mobile can integrate C/C++ code and Apple's iOS uses Objective-C). Nevertheless, in case of Android any error in C that is **not** handled crashes the application before the error is propagated to Java. This security/stability issue should be kept in mind when developing and testing the libraries.

The measurements (see chapter 6) show that GBA Digest bootstrapping

(with channel binding) could be performed in almost the same amount of time as GBA bootstraping, thereby protecting the bootstrapping operation from a Man in the Middle (MitM) attack. The measurements of various operations enable us to conclude that less than 500 ms of additional time during the call-setup and less than 500 μs of overhead is added for every 160 bytes of voice data, but the results are a secure and private VoIP call. Therefore the end user experience should not be affected by introducing VoIP security in the current generation of mobile communication devices.

7.3 Future Work

Currently, the Intiator of a call includes a MIKEY-TICKET `TRANSFER_INIT` message in the SDP of SIP INVITE message and the Responder includes a `TRANSFER_RESP` message in the SDP of SIP 200 OK message (see table 5.1 on page 44). If the Initiator finds the `TRANSFER_RESP` message is invalid, then it hangs up the call. This makes the Responder's UE susceptible to **ghost ringing** (see report [53] and section 3.8). Future work should avoid ghost ringing by using a SIP Session Progress (SIP 183) message and Reliable Provisional Responses [87]. It is also recommended to standardized this way of transferring the key management messages in 3GPP TS 33.328 [4].

3GPP TR 33.914 [60] proposes to bind the TLS tunnel and the SIP digest protocol in order to prevent MitM attacks. However, the MWSB library [78] that was used in this thesis project, for communication between UE and BSF, does not support TLS at this time. Therefore, the reference implementation of 3GPP TR 33.914 used a hard coded string as the `TLS_MK_Extr` (see section 5.3 on page 45). Future work should use TLS between UE and BSF, and find and/or extend a suitable TLS stack in which the TLS tunnel session key could be extracted.

The assumptions made in section 5.3 on page 45 and section 5.4 on page 46 by the author should also be properly defined in 3GPP TR 33.914.

References

[1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711 (Proposed Standard), March 2004. Updated by RFC 5506.

[2] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. RFC 3830 (Proposed Standard), August 2004. Updated by RFC 4738.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761, 6051, 6222.

[4] 3GPP. IP Multimedia Subsystem (IMS) media plane security. TS 33.328, 3rd Generation Partnership Project (3GPP), December 2010.

[5] The Swedish Post and Telecom Agency. PTS Statistics Portal. `http://www.statistics.pts.se/start_en/` Accessed February 27, 2011.

[6] Implementation Specifications for Nokia S60 VoIP v1.4. `http://sw.nokia.com/id/f933e423-ae06-48f3-8612-705fbc576c17/Implementation_Specifications_for_Nokia_S60_VoIP_v1_4_en.pdf`. Accessed February 2, 2011.

[7] VOIP Support in Nokia Devices. `http://wiki.forum.nokia.com/index.php/VoIP_support_in_Nokia_devices`. Accessed February 2, 2011.

[8] Android 2.3 Platform Highlights. `http://developer.android.com/sdk/android-2.3-highlights.html`. Accessed February 2, 2011.

[9] In-Stat Market Research and Intelligence. `http://www.instat.com/` Accessed February 27, 2011.

[10] Wireshark - Network Protocol Analyzer. `http://www.wireshark.org/`. Accessed February 27, 2011.

[11] Romanidis Evripidis. Lawful Interception and Countermeasures: In the era of Internet Telephony. Master's thesis, KTH, September 2008. `http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080922-Romanidis_Evripidis-with-cover.pdf` Accessed March 7, 2011.

[12] Skype Internet Telephony. http://www.skype.com/. Accessed February 27, 2011.

[13] Tom Berson. Skype security evaluation. http://skype.com/security/files/2005-031%20security%20evaluation.pdf, October 2005.

[14] Yahoo! Messenger. http://messenger.yahoo.com/. Accessed February 27, 2011.

[15] Google Chat. http://www.google.com/talk/. Accessed February 27, 2011.

[16] Jumblo, VoIP service provider. http://www.jumblo.com/. Accessed February 27, 2011.

[17] Gerald Q. Maguire Jr. Practical Voice Over IP (VoIP): SIP and related protocols. http://www.ict.kth.se/courses/IK2554/VoIP-20100826.pdf. Accessed February 27, 2011.

[18] International Telecommunication Union. Packet-based Multimedia Communications Systems. ITU-T Recommendation H.323, December 2009. http://www.itu.int/rec/T-REC-H.323-200912-I Accessed February 27, 2011.

[19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141.

[20] F. Andreasen and B. Foster. Media Gateway Control Protocol (MGCP) Version 1.0. RFC 3435 (Informational), January 2003. Updated by RFC 3661.

[21] The Internet Engineering Task Force (IETF). http://www.ietf.org/. Accessed February 27, 2011.

[22] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[23] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006.

[24] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551 (Standard), July 2003. Updated by RFC 5761.

[25] F. Andreasen, M. Baugher, and D. Wing. Session Description Protocol (SDP) Security Descriptions for Media Streams. RFC 4568 (Proposed Standard), July 2006.

[26] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Unicast Secure RTP. RFC 6189 (Informational), April 2011.

[27] J. Mattsson and T. Tian. MIKEY-TICKET: Ticket-Based Modes of Key Distribution in Multimedia Internet KEYing (MIKEY). RFC 6043 (Informational), March 2011.

[28] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113.

[29] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006. Updated by RFC 5746.

[30] D. McGrew and E. Rescorla. Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764 (Proposed Standard), May 2010.

[31] J. Fischl, H. Tschofenig, and E. Rescorla. Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS). RFC 5763 (Proposed Standard), May 2010.

[32] J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). RFC 4474 (Proposed Standard), August 2006.

[33] 3GPP. IP Multimedia Subsystem (IMS); Stage 2. TS 23.228, 3rd Generation Partnership Project (3GPP), September 2010.

[34] Ericsson. Introduction to IMS. `http://www.facweb.iitkgp.ernet.in/~pallab/mob_com/Ericsson_Intro_to_IMS.pdf`. Accessed May 27, 2011.

[35] A. Niemi, J. Arkko, and V. Torvinen. Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA). RFC 3310 (Informational), September 2002.

[36] 3GPP. Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA). TS 33.220, 3rd Generation Partnership Project (3GPP), June 2010.

[37] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588 (Proposed Standard), September 2003. Updated by RFCs 5729, 5719.

[38] 3GPP. Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details. TS 24.109, 3rd Generation Partnership Project (3GPP), June 2010.

[39] 3GPP. Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS). TS 33.222, 3rd Generation Partnership Project (3GPP), June 2010.

[40] 3GPP. Generic Authentication Architecture (GAA); Support for subscriber certificates. TS 33.221, 3rd Generation Partnership Project (3GPP), June 2010.

[41] 3GPP. Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Stage 3. TS 29.109, 3rd Generation Partnership Project (3GPP), April 2010.

[42] I. Abad. Secure Mobile VoIP. Master's thesis, KTH, June 2003. http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/030626-Israel_Abad_Caballero-final-report.pdf Accessed January 27, 2011.

[43] Minisip - SIP User Agent. http://minisip.org/. Accessed February 27, 2011.

[44] J. Bilien. Key Agreement for secure Voice over IP. Master's thesis, KTH, December 2003. http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/031215-Johan-Bilien-report-final-with-cover.pdf Accessed January 27, 2011.

[45] Johan Bilien, Erik Eliasson, and Jon-Olov Vatn. Call establishment delay for secure voip. In *Proceedings of WiOpt '04: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2004. http://www.minisip.org/publications/secvoip.pdf Accessed January 27, 2011.

[46] A.L. Alexander, A.L. Wijesinha, and R. Karne. An Evaluation of Secure Real-Time Transport Protocol (SRTP) Performance for VoIP. In *Network and System Security, 2009. NSS '09. Third International Conference on*, pages 95 –101, October 2009.

[47] snom technology AG - VoIP phones. http://www.snom.com/. Accessed February 27, 2011.

[48] Twinkle - SIP softphone for linux. http://www.xs4all.nl/~mfnboer/twinkle/index.html. Accessed February 27, 2011.

[49] Gholam H. Khaksari, Alexander L. Wijesinha, Ramesh K. Karne, Long He, and Sandeep Girumala. In *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE, title=A Peer-to-Peer Bare PC VoIP Application*, pages 803 –807, 2007.

[50] O. Olsson. Security Analysis of Key-Exchange Protocol Mikey-Ticket. Master's thesis, KTH, 2010. http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2010/rapporter10/olsson_oscar_10019.pdf Accessed May 27, 2011.

[51] Johan Bilien, Erik Eliasson, and Jon-Olov Vatn. Experiences of using a secure VoIP user agent on PDAs, May 2004. http://www.minisip.org/publications/wifi-voice-minisip.ppt Accessed January 27, 2011.

[52] Johan Bilien, Erik Eliasson, Joachim Orrblad, and Jon-Olov Vatn. Secure voip : call establishment and media protection. In *Proceedings of 2nd Workshop on Securing Voice over IP*, 2005. `http://www.minisip.org/publications/secvoip-minisip-camera.pdf` Accessed January 27, 2011.

[53] Erik Eliasson. Secure VoIP performance on handheld devices. Technical Report 06:03, KTH, Telecommunication Systems Laboratory, TSLab, 2006. Paper D on `http://www.minisip.org/publications/ErikEliasson_LicentiateThesis.pdf` Accessed February 1, 2011.

[54] Erik Eliasson. Secure Internet telephony : design, implementation and performance measurements, 2006. `http://www.minisip.org/publications/ErikEliasson_LicentiateThesis.pdf` Accessed January 27, 2011.

[55] J. Orrblad. Alternatives to MIKEY/SRTP to secure VoIP. Master's thesis, KTH, March 2005. `http://www.minisip.org/publications/Thesis_Orrblad_050330.pdf` Accessed January 28, 2011.

[56] Tomas Joelsson. Mobile Web Browser Extensions. Master's thesis, KTH, April 2008. `http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080412-Tomas_Joelsson-with-cover.pdf` Accessed February 27, 2011.

[57] Sing Li and Jonathan Knudsen. *Beginning J2ME: From Novice to Professional*. APress, third edition, 2005.

[58] J. Arkko, F. Lindholm, M. Naslund, K. Norrman, and E. Carrara. Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP). RFC 4567 (Proposed Standard), July 2006.

[59] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), October 2006.

[60] 3GPP. SSO for Application Security for IMS - based on SIP Digest. TR 33.914, 3rd Generation Partnership Project (3GPP), May 2011.

[61] 3GPP. IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents. TS 29.228, 3rd Generation Partnership Project (3GPP), April 2011.

[62] 3GPP. Cx and Dx interfaces based on the Diameter protocol; Protocol details. TS 29.229, 3rd Generation Partnership Project (3GPP), September 2010.

[63] Sipdroid - Open source SIP/VoIP client for Android. `http://sipdroid.org/`. Accessed February 27, 2011.

[64] Nebula Android - SIP based application for instant group communication. `http://csd.xen.ssvl.kth.se/csdlive/content/8-yards`. Accessed February 27, 2011.

[65] SIP Communicator - the Java VoIP and Instant Messaging client. http://sip-communicator.org/. Accessed February 28, 2011.

[66] The Zfone Project. http://zfoneproject.com/. Accessed February 28, 2011.

[67] LibSRTP - a library for secure RTP. http://srtp.sourceforge.net/srtp.html. Accessed February 28, 2011.

[68] ZORG - ZRTP implementation. http://www.zrtp.org/. Accessed February 28, 2011.

[69] Gartner - Information technology research and advisory company. http://www.gartner.com/. Accessed February 28, 2011.

[70] Socks. http://www.eurescom.eu/~public-webspace/P1000-series/P1009/doc3_3.html Accessed March 7, 2011.

[71] DroidWall - Android Firewall. http://code.google.com/p/droidwall/. Accessed March 7, 2011.

[72] J.C. Mogul and S.E. Deering. Path MTU discovery. RFC 1191 (Draft Standard), November 1990.

[73] P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese. IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines. RFC 3580 (Informational), September 2003.

[74] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov. TCP over Second (2.5G) and Third (3G) Generation Wireless Networks. RFC 3481 (Best Current Practice), February 2003.

[75] IMSDrouid - Open source 3GPP IMS Client for Android. http://code.google.com/p/imsdroid/. AccessedMay 27, 2011.

[76] 3GPP. IP Multimedia Subsystem (IMS) media plane security. TR 33.828, 3rd Generation Partnership Project (3GPP), June 2010.

[77] Evolutionary Prototyping White Paper. http://www.construx.com/File.ashx?cid=814. Accessed May 27, 2011.

[78] Mobile Web Security Bootstrap API. https://labs.ericsson.com/apis/mobile-web-security-bootstrap/. Accessed May 27, 2011.

[79] FOKUS Home Subscriber Server. http://www.openimscore.org/project/FHoSS. Accessed May 27, 2011.

[80] J. Arkko, F. Lindholm, M. Naslund, K. Norrman, and E. Carrara. Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP). RFC 4567 (Proposed Standard), July 2006.

[81] OpenSSL for Android. https://github.com/eighthave/openssl-android. Accessed May 27, 2011.

[82] cURL library. http://curl.haxx.se/. Accessed May 27, 2011.

[83] GlassFish - Open Source Java EE Application Server. http://glassfish.java.net/. Accessed May 27, 2011.

[84] Apache Commons Codec. http://commons.apache.org/codec/. Accessed May 27, 2011.

[85] Javax Patch for Android. http://grepcode.com/snapshot/repo1.maven.org/maven2/it.tidalwave.bluebill/it-tidalwave-android-javax-xml-datatype/1.0.6. Accessed May 27, 2011.

[86] Profiling Android Applications. http://developer.android.com/guide/developing/debugging/debugging-tracing.html. Accessed February 27, 2011.

[87] J. Rosenberg and H. Schulzrinne. Reliability of Provisional Responses in Session Initiation Protocol (SIP). RFC 3262 (Proposed Standard), June 2002.

Appendix A

Message Flows

This appendix illustrates the message flows between various components as captured by Wireshark. Note that the message flows correspond to the GBA Digest bootstrapping.

A.1 Between UE and BSF during Bootstrapping

The bootstrapping request/response message flows look like the following.

```
GET /bsfv2/bsf HTTP/1.1
Authorization: Digest username="initiator@192.36.158.101",realm
    ="192.36.158.101",nonce="",uri="/bsfv2/bsf",response=""
Host: 192.36.158.101:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

HTTP/1.1 401 Unauthorized
X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server Open Source
    Edition 3.1 Java/Sun Microsystems Inc./1.6)
Server: GlassFish Server Open Source Edition 3.1
Set-Cookie: JSESSIONID=ba4847f10c283c5803b24dd1b854; Path=/bsfv2;
    HttpOnly
WWW-Authenticate: Digest realm="localhost",nonce="LvnKbf09xY76Vlk/2
    OnftxANVbjEymwldtkbFhUt8rk=",algorithm="MD5",qop="auth"
Content-Length: 0
Date: Thu, 26 May 2011 09:31:00 GMT

GET /bsfv2/bsf HTTP/1.1
Authorization: Digest username="initiator@192.36.158.101",realm="
    localhost",nonce="LvnKbf09xY76Vlk/2OnftxANVbjEymwldtkbFhUt8rk=",
    uri="/bsfv2/bsf",qop="auth",nc=00000001,cnonce="
    J9i0yjbYo8dQqW2JJHqxYw==",response="58
    cfd7b9c34a0b5f13fcf0ad74019a54",bres="
    b089ce901a600788c5382043dba42411",algorithm="MD5"
Host: 192.36.158.101:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
Cookie: JSESSIONID=ba4847f10c283c5803b24dd1b854
```

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server Open Source
    Edition 3.1 Java/Sun Microsystems Inc./1.6)
Server: GlassFish Server Open Source Edition 3.1
Authentication-Info: qop="auth",rspauth="03
    c0ed50aae913ed8c3f8a876a4711be",cnonce="J9i0yjbYo8dQqW2JJHqxYw
    ==",nc=00000001
Content-Type: application/vnd.3gpp.bsf+xml
Content-Length: 221
Date: Thu, 26 May 2011 09:31:00 GMT

<?xml version="1.0" encoding="UTF-8"?>
<BootstrappingInfo xmlns="uri:3gpp-gba-digest"><btid>LvnKbf09xY76Vlk
    /2OnftxANVbjEymwldtkbFhUt8rk=@localhost</btid><lifetime
    >2011-05-26T10:31:00.228Z</lifetime></BootstrappingInfo>
```

A.2 Between BSF and HSS during Bootstrapping of UE

The BSF gets SD-AV for the initiator from HSS as following. Note that `Auth-Application-Id` is chosen for test by the author.

```
MAR:    Command Code: 303 Multimedia-Auth
        Session-Id: localhost;-365155413;55
        Vendor-Id: 10415
        Auth-Application-Id: 16777228
        Auth-Session-State: NO_STATE_MAINTAINED (1)
        Origin-Host: localhost
        Origin-Realm: localhost
        Destination-Realm: localhost
        Destination-Host: localhost
        User-Name: initiator@192.36.158.101

MAA:    Command Code: 303 Multimedia-Auth
        Session-Id: localhost;-365155413;55
        Vendor-Id: 10415
        Auth-Application-Id: 16777228
        Result-Code: DIAMETER_SUCCESS (2001)
        Auth-Session-State: NO_STATE_MAINTAINED (1)
        Origin-Host: localhost
        Origin-Realm: localhost
        User-Name: initiator@192.36.158.101
        SIP-Number-Auth-Items: 1
        SIP-Item-Number: 1
        SIP-Authentication-Scheme: SIP Digest
        Digest-HA1: 6a1b6417d55f28734b1755babef7f980
        Digest-Realm: localhost
        GBA-UserSecSettings: 3
            C3F786D6C2076657273696F6E3D22312E302220656E636F...
```

A.3 Between Initiator's UE and KMS

The `REQUEST_INIT/RESP` messages look like the following.

```
POST /keymanagement?requesttype=ticketrequest HTTP/1.1
Host: 192.36.158.101:8080
Content-Type: application/mikey
Content-Length: 360
User-Agent: KMS_Test
From: initiator@192.36.158.101
Date: Thu, 26 May 2011 09:30:52 GMT

AQcFgAAAAAAAAQsA1Ur1PAAAAAAX......


HTTP/1.1 200 OK
Date: Thu, 26 May 2011 09:31:00 GMT
Content-Type: application/mikey
Content-Length: 436

AQkFAAAAAAAAARkA1Ur1PAAAAAABAEoAAAAA......
```

A.4 Between KMS and BSF during Bootstrapping Usage

The bootstrapping usage messages look like the following.

```
POST /bsfv2/requestBootstrappingInfo HTTP/1.1
Content-Type: text/xml; charset=utf-8
User-Agent: Java/1.6.0_24
Host: 192.36.158.101:8080
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 303

<?xml version="1.0" encoding="UTF-8"?>
<requestBootstrappingInfoRequest xmlns="http://www.3gpp.org/
    GBAService"><btid>LvnKbf09xY76Vlk/2OnftxANVbjEymwldtkbFhUt8rk=
    @localhost</btid><nafid>MTkyLjM2LjE1OC4xMDE6ODA4MAEAAAAA</nafid><
    gsid>0</gsid><gbaUAware>True</gbaUAware></
    requestBootstrappingInfoRequest>


HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server Open Source
    Edition 3.1 Java/Sun Microsystems Inc./1.6)
Server: GlassFish Server Open Source Edition 3.1
Content-Type: text/xml;charset=utf-8
Transfer-Encoding: chunked
Date: Thu, 26 May 2011 09:31:00 GMT

238
<?xml version="1.0" encoding="UTF-8"?>
<requestBootstrappingInfoResponse xmlns="http://www.3gpp.org/
```

```
    GBAService"><impi>initiator@192.36.158.101</impi><meKeyMaterial>
    mLJbLxgP+FE6pN1ky++5r+4Wpz6qjE0n9cpWdL5kxtw=</meKeyMaterial><
    uiccKeyMaterial>5u7xPk9Dy666oMsNaed99ESeEciHQTDSuQJWHJ94npY=</
    uiccKeyMaterial><keyExpiryTime>2011-05-26T10:31:00.228Z</
    keyExpiryTime><bootstrappingInfoCreationTime>2011-05-26T09
    :31:00.228Z</bootstrappingInfoCreationTime><ussList><![CDATA[<?
    xml version="1.0" encoding="UTF-8"?>
<ussList />
]]></ussList></requestBootstrappingInfoResponse>

0
```

A.5 Between Initiator's UE and Responder's UE during Initiation of a Call

The SIP INVITE message contains the MIKEY-TICKET `TRANSFER_INIT` message in the SDP offer as the following. Notice the `key-mgmt` attribute.

```
INVITE sip:responder@192.36.158.78:33867;transport=udp SIP/2.0
Record-Route: <sip:192.36.158.101;lr=on>
Via: SIP/2.0/UDP 192.36.158.101;branch=z9hG4bKcfc7.4b545663.0
Via: SIP/2.0/UDP 192.36.158.107:33785;received=192.36.158.107;rport
    =33785;branch=z9hG4bK99633
Max-Forwards: 69
To: <sip:responder@192.36.158.101>
From: <sip:initiator@192.36.158.101>;tag=z9hG4bK29389008
Call-ID: 143839269191@192.36.158.107
CSeq: 1 INVITE
Contact: <sip:initiator@192.36.158.107:33785;transport=udp>
Expires: 3600
User-Agent: Sipdroid/2.1 beta/Nexus One
Content-Length: 859
Content-Type: application/sdp

v=0
o=initiator@192.36.158.101 0 0 IN IP4 192.36.158.107
s=Session SIP/SDP
c=IN IP4 192.36.158.107
t=0 0
a=key-mgmt:mikey
    AQoFgAAAAAABAAAAAAAAAAAAAAsA1Ur1PAAAAAAAXEAAAAAAAAAAA.....
m=audio 21000 RTP/SAVP 9 8 0 97 3 106 101
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:9 G722/8000
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
m=video 21070 RTP/SAVP 103
a=rtpmap:103 h263-1998/90000
```

A.6 Between Responder's UE and KMS

The `RESOLVE_INIT/RESP` messages look like the following.

```
POST /keymanagement?requesttype=ticketresolve HTTP/1.1
Host: 192.36.158.101:8080
Content-Type: application/mikey
Content-Length: 464
User-Agent: KMS_Test
From: responder@192.36.158.101
Date: Thu, 26 May 2011 09:30:36 GMT

AQwFgHTwpnsAAQsA1Ur1LAAAAAAAXEG+lZewAAAAA..... (truncated)

HTTP/1.1 200 OK
Date: Thu, 26 May 2011 09:31:00 GMT
Content-Type: application/mikey
Content-Length: 196

AQ4FgHTwpnsAAQEA1Ur1LAAAAAALAQBRew7n0/8XDddqOK..... (truncated)
```

A.7 Between Responder's UE and Initiator's UE during Acceptance of a Call

The SIP 200 OK contains the MIKEY-TICKET `TRANSFER_RESP` message as the following. Notice the `key-mgmt` attribute. Note that the `TRANSFER_RESP` can be sent in other SIP messages as well.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.36.158.101;branch=z9hG4bKcfc7.4b545663.0
Via: SIP/2.0/UDP 192.36.158.107:33785;received=192.36.158.107;rport=33785;
    branch=z9hG4bK99633
Record-Route: <sip:192.36.158.101;lr=on>
To: <sip:responder@192.36.158.101>;tag=484bf542a61edbc4
From: <sip:initiator@192.36.158.101>;tag=z9hG4bK29389008
Call-ID: 143839269191@192.36.158.107
CSeq: 1 INVITE
Contact: <sip:responder@192.36.158.78:33867;transport=udp>
Server: Sipdroid/2.1 beta/Nexus One
Content-Length: 401
Content-Type: application/sdp

v=0
o=responder@192.36.158.101 0 0 IN IP4 192.36.158.78
s=Session SIP/SDP
c=IN IP4 192.36.158.78
t=0 0
a=key-mgmt:mikey AQsFgAAAAAABAAAAAAAAAAAAAAsA1Ur1PAAAAAA..... (truncated)
m=audio 21000 RTP/SAVP 9 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
m=video 21070 RTP/SAVP 103
a=rtpmap:103 h263-1998/90000
```

TRITA-ICT-EX-2011:111

www.kth.se