Master Thesis

Security in VoIP-Telephony Systems

Johann Thalhammer

INSTITUTE FOR APPLIED INFORMATION PROCESSING AND COMMUNICATIONS

GRAZ UNIVERSITY OF TECHNOLOGY

Head: O.Univ.-Prof., Dipl.-Ing. Dr.techn. Reinhard Posch

Examinor:
O.Univ.-Prof.DI. Dr.techn. Reinhard Posch

Department:
IP-Telephony

Supervisor:
DI. Herbert Leitold

Supervisor:
DI. Wolfgang Platzer

The spread of local area networks within companies and the possibility of voice transmission over the Internet support the evolution of VoIP Telephony Systems. The obvious next step is to offer extensive VoIP Telephony Service on the Internet. The distribution of the system’s elements on the "Net" introduces security holes. These arise through the accessibility of the interfaces and protocols. All of a sudden security aspects get as important as Quality of Service (QoS) and Value Added Services. This thesis, in cooperation with Infonova, deals especially with security aspects regarding H.323 (ITU-T Recommendation for Packed-Based Multimedia Communications Systems) and implements parts of the results in their product VoIP-Center.

Contents

Chapter 1

Introduction

VoIP telephony or IP telephony is the transportation of voice traffic over the Internet protocol (IP). The Internet is an interconnection between networks that use IP. Over the years the Internet has become a basis for applications and services it was not intended for at the beginning. It grew to a market with high potentials especially for services such as IP telephony. This is due to three reasons: Firstly, telephony is a business with high revenues and a lot of customers. Secondly, more and more people know the Internet and use it in their daily life. Thirdly, the Internet with its flexibility, fast development and openness will generate many new services [14].

A major issue in the Internet is security. A lot of security incidents occurred in the last years and their number is still increasing [19]. Therefore security is an essential topic for VoIP telephony systems.

1.1 Project description

This project deals with the security of a VoIP telephony system. Figure 1.1 shows the architecture of such a system. The involved components are endpoints (EP), gatekeeper (GK) and the back-end service (BES). The EPs are the devices that enable a user of the system to call another person. The GK administers the EPs and the BES stores important information that allows the GK to connect two parties with each other. The underlying protocol of this scenario is H.323, a standard for IP telephony.

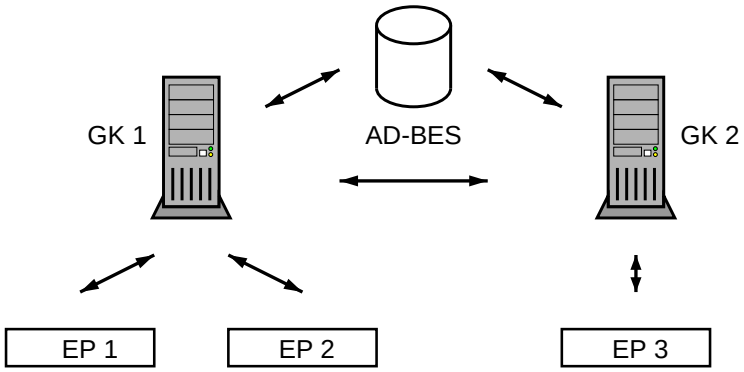

Figure 1.1: VoIP architecture

The project was done in cooperation with INFONOVA GmbH. Its goal was to analyse the protocols used between the entities participating in the IP telephony system, try to find weaknesses and propose security improvements. The work consists of two parts, a theoretical one and a practical one. The theoretical part explains the technologies behind IP telephony and security, analyses the protocol applied in the VoIP architecture of INFONOVA and proposes a security framework. The practical work was the implementation of a part of the result in the VoIP center of INFONOVA.

1.2 Structure of the thesis

This thesis is composed of four parts: the explanation of the technologies IP telephony is based on, a general approach to security for information systems connected to the Internet, the analysis of a telephone call scenario including the proposal of a security framework and the description of the part that was implemented.

The underlying technologies for IP telephony and security are explained in chapter 2 "*Introduction to TCP/IP*", chapter 3 "*IP Telephony*" and chapter 5, "*Cryptography*". The chapter Introduction to TCP/IP describes basics about the Internet and the protocols used to be able to implement IP telephony. Chapter IP Telephony covers the IP telephony protocols H.323 and SIP. Cryptography contains necessary terms, algorithms and protocols necessary in the following chapters. Chapter 4 "*Secure Information Systems*" deals with security of an information system on the Internet. The security analysis of the IP telephony architecture of INFONOVA as well as their results follow in chapter 6 "*Security Concept*". The practical part of this project is presented in chapter 7 "*Implementation*". It describes the part of the analysis that was added to the VoIP telephony center of INFONOVA GmbH. The end of this thesis builds chapter 8 "*Conclusion*", which discusses the project, its results and areas for further research.

Chapter 2

Introduction to TCP/IP

The Internet started in 1973 as a research project of the U.S. Defense Advanced Research Projects Agency (DARPA). The objective was to develop protocols to allow computers to communicate across many connected networks. The system of protocols is known as the TCP/IP-suite. It was named after the initial two developed protocols the transmission control protocol (TCP) and the Internet protocol (IP). This protocol suite is the basis for many applications and services such as voice transmission over the Internet. This chapter gives background information about networking in general and the most important protocols that facilitate VoIP. This includes the two models for a network architecture, the ISO-OSI model and the TCP/IP model. The introduced protocols are IP, user datagram protocol (UDP), TCP and real-time transport protocol (RTP).

2.1 Network architecture

Computer networks are very complex constructions. The connectivity of computers requires a lot of complicated software. To ease the implementation of application programs an abstraction layer was introduced. Programs that want to access network functions do that via an interface. The network architecture is typically separated into layers. Each layer represents a different aspect of the system. The communication with other layers is done via interfaces.

Figure 2.1 depicts an example for a layered structure. The lowest layer is the hardware layer. It is the basis for computer networks. Everything connected to hardware belongs to this layer. The next layer provides connectivity between hosts. It implements the functionality of sending data between two computers. It provides an interface to access its services and uses the hardware layer for its operation. The communication of two applications on different hosts requires some kind of channel between the applications' processes. This functionality forms another layer in this

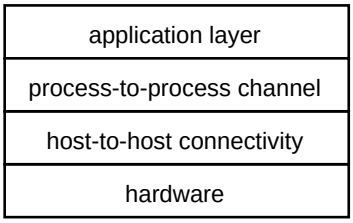

Figure 2.1: Example for a layered network architecture

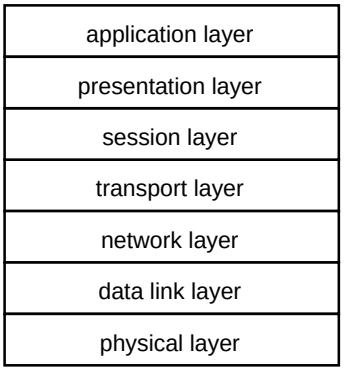

Figure 2.2: OSI network model

example. It utilizes the connection of hosts and offers its functions to the applications on the top of this architecture.

There are two established models for network architectures. The standard model for networking protocols and distributed applications is the model from the International Organization for Standardization (ISO). However, the model used in the Internet is the TCP/IP protocol suite [43].

2.1.1 The OSI model

The open system interconnection (OSI) is the model of the ISO. It is specified in standard ISO/IEC 7498-1 [21]. The network functions are seperated into seven layers. The functionality of each layer can be implemented by one or more protocols. The OSI model intends to be a reference model for the implementation of protocols.

Figure 2.2 illustrates the OSI model. The first and lowest layer is the *physical* layer. It defines the physical medium on which the data is transmitted and is responsible for the transmission of raw bits over communication links. The next higher layer, the second layer, is the *data link* layer. It collects the raw bits sent to so called frames. These frames are the entities exchanged between two adjacent hosts. The network adaptor of a computer and its driver software can be seen as its implementation. The third layer of the OSI model is called the *network* layer. The network layer masks the differences of different sub networks to provide a consistent network service. The data units exchanged on that layer are called packets. The network layer is responsible for the end-to-end delivery of those packets. The process of delivering packets over networks is also referred to as routing. Every node (host) on the Internet must implement these three layers.

The layer on top of the network layer is the *transport* layer. It is the fourth layer in the network stack. It provides, what was called process-to-process channel in the example of figure 2.1. This is a transparent way of transmitting data. It relieves the entity using the service of the transport layer of any concern regarding the details of the data transmission. The exchanged data units on this layer are called messages.

There is some disagreement about the layers on top of the transport layer, the *session* layer, the *presentation* layer and the *application* layer. According to the standard the session layer, the fifth layer, provides a *session-connection* between two entities. A session is in charge of all the transport streams that are assigned to the session. For example a videoconference session consists of two transport streams, one for audio and one for video signals. The presentation layer provides means for representation of data transferred between applications. This can be for instance syntax

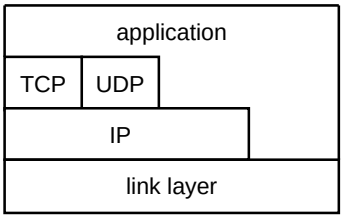

Figure 2.3: TCP/IP protocol architecture

independency or whether an integer is 16, 32 or 64 bit long [43]. Thus, the service of the presentation layer is the negotiation of the syntax that is used to transfer the data between two systems and providing the interface to access its functions.

Finally, the last layer of the model is the application layer. It includes all those functions that make a communication between two systems necessary. This includes functions performed by programs or human beings themselves.

2.1.2 TCP/IP network model

The TCP/IP model is the network architecture of the Internet. Its basis is the experience with an earlier network, the *ARPANET*. The ARPANET as well as the Internet was funded by the DARPA. The knowledge gained with the ARPANET and the Internet had a lot of influence on the development of the OSI architecture. Figure 2.3 illustrates the TCP/IP model. Its layers are not as strictly divided as the ones of the OSI model. It is possible for applications to access the services of every layer. The lowest layer in this model is the link layer. It consists of many different network protocols. A comparison with the OSI model shows that the functionality of the network layer in the TCP/IP model corresponds to layer one, the physical layer, and two, the data link layer. It describes the deployed hardware of the network as well as the network adaptors and their device driver. The second layer consists of IP. It complies to the network layer of the OSI model. Its functionality is to provide a transparent network with one interface. The subnetworks can consist of different network technologies with different protocols. The third layer of the TCP/IP suite is the equivalent to the transport layer, layer number four, of the OSI model. There are two different protocols that form this layer, UDP and TCP. They provide alternative logical channels to applications. TCP offers a reliable connection that can be seen as a byte-stream channel, whereas UDP supplies a unreliable datagram service. Bytes sent to a TCP stream arrive at the other end of the channel in the order they were sent. The application must not care about data loss. The service of UDP sends its datagrams (individual messages) without any guarantee of delivery. The last layer is the application layer. It contains the functionality of the application.

2.1.3 Encapsulation

Encapsulation describes the process of building the actual frame that is sent over the network out of the data an application wants to transmit. Additional information is necessary to send application data to another host. To explain encapsulation an application is assumed that resides on the transport layer of the TCP/IP suite and uses TCP as transport protocol. The frame on the link layer is an Ethernet frame. Ethernet is one possible technology to implement the link layer.

Figure 2.4 demonstrates encapsulation. As already mentioned in the previous subsection the

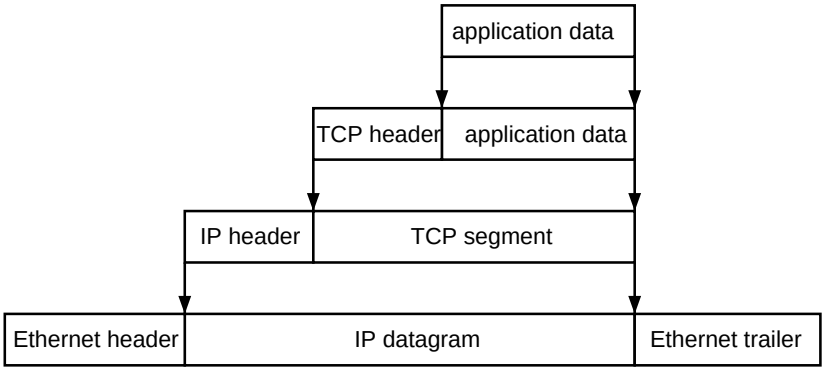

Figure 2.4: Encapsulation of user data into an Ethernet frame

application can also skip the transport layer and directly access the service offered by the link layer.

The data sent by the application has to pass the protocol stack, layer by layer. The application hands over the data to the transport layer. Here, this is a TCP connection. The data passed to a layer is called the payload of the protocol. The transport layer adds its layer specific data, the TCP header to the payload and forwards the packet to the next layer, the IP layer. A TCP packet is also called a *TCP segment*. IP appends its header data to the TCP segment to form an IP datagram and passes it to the link layer. The link layer protocol (here Ethernet) builds the final frame and sends it to the destination.

In most cases header data is added in the beginning of the packet. However, some protocols add data at the beginning and at the end of the payload. Data added at the beginning is called a *header* and data appended to the payload is called a *trailer*. Every layer has its own header data. It hands this together with its payload over to the next layer. The packet reaches its final length at the link layer. At this point it is composed of the header data (and tailer data) of every layer and the applications data. If the length of the entire packet exceeds the maximum transmission unit (MTU) of the link layer the packet is separated into fragments. This is done by IP on the network layer. The final packets are sent to the destination over the network. There again all the stages of the protocol stack have to be gone through. This time it is done backwards, starting at the link layer. The link layer receives the packet, removes its header and passes its payload to the network layer. This process is the same in the network layer as well as in the transport layer. Finally, the application at the destination host receives the data.

2.2 Internet protocol

The main protocol of the TCP/IP suite is the Internet protocol (IP). It is the implementation of the network layer and is officially specified in RFC791 [15]. IP is designed for the use in interconnected computer networks. It provides an unreliable host-to-host packet delivery service. The data packets are also called IP datagrams . The term unreliable means that there are no guarantees that the IP datagram reaches its destination in the right order. It even cannot be ensured that the datagrams actually arrive at the destination.

The two main properties of IP are addressing and fragmentation. The information required to provide these services is stored in the header of the IP datagram. Figure 2.5 shows the structure of

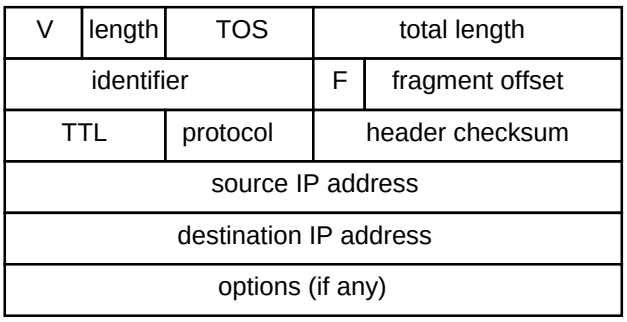

Figure 2.5: Header of an IP packet

the IP header. The remainder of this section explains the fields of the header that are necessary to understand the main functionality. A more detailed description of it can be looked up in [48].

2.2.1 Addressing

The addresses are used to deliver IP datagrams from the source to the destination. Every datagram needs a *source address* and a *destination address*. The corresponding fields in the header of the IP datagram carry the same names. IP addresses are 32 bits long. It is possible to address about four billion hosts with this address space. An IP address is written as four byte values separated by dots, e.g. $10.1.13.105$. The reason for this representation is the hierarchical organisation of the Internet. Each IP address consists of a network part and a host part. A network can be a connection of one or more subnetworks. All hosts that belong to one network (and subnetwork) must have the same network part of the address. The classification of different networks is depicted in figure 2.6. There are five different classes. They are identified by the values of the first four most significant bits. The different categories define the length of the network part and the host part of the IP address. They specify also the number of (sub)networks and the number of hosts per network. In class A for example the network part is seven bits and the host part is 24 bits long.

2.2.2 Fragmentation

The theoretical maximum size of an IP datagram is 65535 bytes (64kByte). IP can be used by different networks with different abilities for the transmission of frames. The maximum length of one frame is given by the MTU. If the length of a packet exceeds the MTU, IP must fragment the packet. The packet is splitted into packets with a size that correspond to the MTU. Then they are delivered separately to the destination. The packets are reassembled at the destination. This can be done via information contained in the header of the IP datagram, the field *flags*. *Flags* is made of three bits. Only two of them are actually used. The first bit has the name *do not fragment*. If this flag is set and the packet is longer than the MTU, the packet must be thrown away. The second bit, *more fragments*, says whether there are other packets with fragments of the original IP datagram or not. A further field that is important for the reconstruction of the original IP datagram is *fragment offset*. It specifies the offset of the first bit of the payload of a fragmented packet in the original packet. This field is 13 bits long. It allows to address 4 kByte of data. Since IP datagrams can be 64kByte long, the offset has to be given in entities of 8 byte.

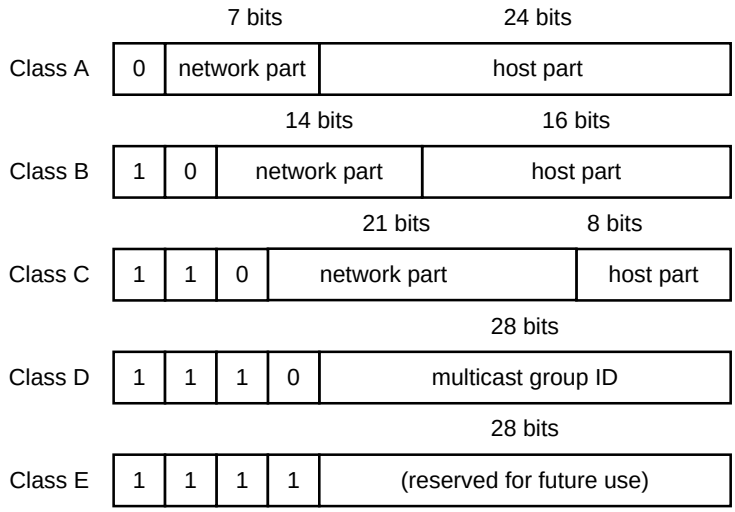

Figure 2.6: Classification of IP addresses

source port	destination port
length	checksum

Figure 2.7: Header of an UDP packet

2.3 UDP - User datagram protocol

This protocol resides in the transport layer and uses IP as network layer protocol. It is specified in RFC768 [44]. As earlier stated IP uses IP addresses to identify individual hosts. Assuming that there are several applications running on one host, the question arises how a transport layer protocol knows which packet is meant for which application? The answer is: port numbers. Port numbers are used to identify different server applications on a host. Transport layer protocols, UDP as well as TCP, which is explained later, have two port numbers in their headers. These identify the source from which the data is sent and its destination (see figure 2.7). The names of these ports are: *source port* and *destination port*.

Other fields in the UDP header are *length* and *checksum*. The length-field specifies the length of the entire datagram: header plus data. The minimum value for the length-field is eight. In fact, this is the length of the UDP header without any data. The checksum is calculated over header and body.

UDP is a datagram oriented service. Each output operation produces one datagram that is sent over the network. Like IP, UDP is an unreliable service.

2.4 TCP - Transmission control protocol

TCP is the second transport layer protocol in the TCP/IP suite. It is specified in RFC793 [16]. TCP provides a continuous data flow between two endpoints on the network. It provides a full

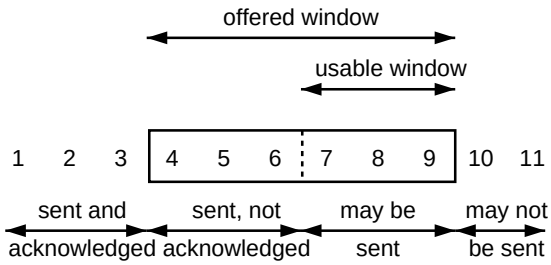

Figure 2.8: Visualization of sliding windows

duplex service to the application layer, which means that data can flow in both directions. For an application it is like a byte stream. The data is sent into the stream assuming that it will arrive at the other end, byte after byte, in the correct order. However, IP neither guarantees packet delivery nor an arrival of data packets in the right order. Therefore, TCP must have its own mechanism. Every TCP packet is equipped with a sequence number that allows the recognition of missing packets, packets received in the wrong order or duplicated packets. TCP splits the data intended to be sent into best sized “chunks”. This unit of information is called a *segment*. For every received segment, an *acknowledge* is sent back to the sender. This confirms the reception of the packet. If the packet does not arrive in a certain time interval (the sender does not receive an acknowledge), the sender retransmits the packet. If packets arrive out of order, they can be reassembled because of the sequence numbers in each TCP packet. The technique that ensures the reliable transmission of data is called *sliding window*. A simplified version is illustrated in figure 2.8. The situation of the receiver is shown above the rectangle of the window, the one of the sender below. The digits in the middle represent the byte stream.

The sender maintains a buffer for the data that has been sent but has not been acknowledged by the receiver. The receiver has a buffer for the data it received. Data that is read is deleted from that buffer. For the receiver it does not matter whether the data arrived in order or out of order. The receiver tells the sender how much data it still can store. This avoids an overflow of the receiver’s buffer. This number is called the window size. It is sent in the packets that acknowledge the receipt of data. The procedure of sending the window size is called *advertising* a window. In the example of figure 2.8 the window size is six. Then the sender knows how many frames it can send without having the receiver throwing packets away. The term sliding refers to the movement of the window in the stream. The byte stream can be imagined as a line. If the number of bytes acknowledged by the receiver increases the begin of the window moves to the right, the window closes. The end of the window moves also to the right if the receiver reads data, acknowledges it and advertises a new window. This is also described as the opening of the window.

2.4.1 TCP header

The header of TCP contains data that is necessary to fulfill its functionality. Figure 2.9 shows the complete header. The first two fields are port numbers, the *destination port* and the *source port* number. These two numbers together with the source and destination address of IP identify the sending and receiving applications. The *sequence number* is the offset number of the first data byte of the payload in the byte stream of the connection. The following field *acknowledgement number* is used to acknowledge received and read data. The meaning of *length* is the header length including the options in bytes. In other words it defines the start of the data in the TCP segment.

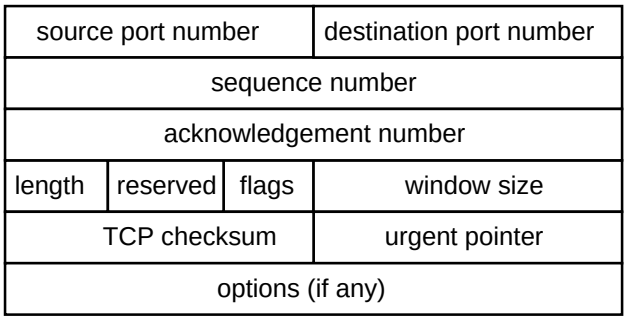

Figure 2.9: Header of a TCP packet

The flag field contains six flags:

- Urgent (URG): It enables urgent mode. One end signals the other end that some form of urgent data has been placed into the stream.
- Acknowledge (ACK): It indicates that the acknowledgement number contained in this TCP packet is valid.
- Push (PSH): It asks the receiver to pass the data to the application as soon as possible.
- Reset (RST): It resets the connection.
- Synchronize (SYN): It is used during the connection establishment to synchronize sequence numbers.
- Finished (FIN): Its sender indicates that it wants to terminate the connection.

The field *window size* is used for the advertisement of the transmission window size. The maximum window size is 64 kByte. TCP maintains a *TCP checksum* that is calculated over header and payload. This assures the integrity of the data. In case of corruption, the packet is discarded and no acknowledge is sent back. If the checksum is corrupted, the packet is discarded. It is even discarded if the data in the packet is correct. The reason is that it is not possible to tell whether the header or the data was changed. The field *urgent pointer* is only valid if the corresponding flag is enabled. The last fields of the header are options, if any are present.

2.4.2 Connection establishment

The protocol for the establishment of a connection is also known as the *three-way handshake protocol*. It is depicted in figure 2.10 and explained below. The showed model for the connection is called the client-server model. Here one process is bound to a port on one host and "listens" for applications connecting to it. These connecting applications are called clients. Clients *request* services from a server. A server *responds* to the requests. In the following connection data can be send in both directions. The establishment of the connection is a three-stage process:

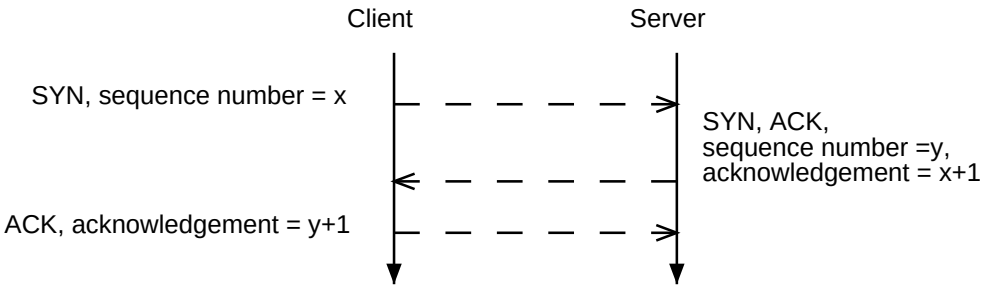

Figure 2.10: Timeline for the three-way-handshake algorithm

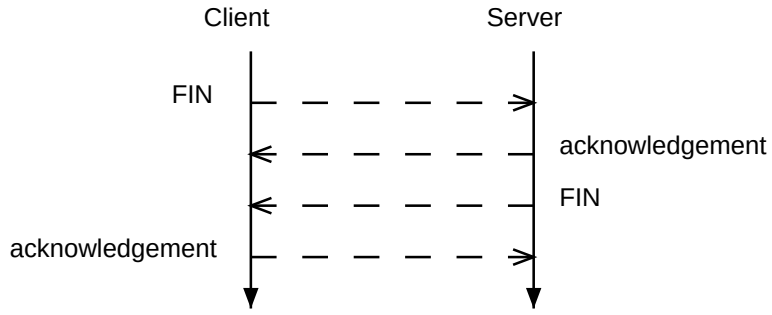

Figure 2.11: Timeline of the termination of a connection

1. The host that wants to establish a connection (the client) sends a TCP segment with the *SYN* flag set. The host chooses an *Initial Sequence Number(ISN)* for the first segment. The sequence number of the first data byte in that stream is the ISN increased by one since the SYN flag consumes one sequence number.

2. The server then responds with its own first segment including a set SYN flag. Additionally the reception of the first packet is acknowledged with the sequence number of the client increased by one. The server chooses the sequence number of this packet.

3. The last segment in this establishment protocol is sent by the client and acknowledges the reception of the segment and the ISN from the server.

The action of transmitting the first SYN segment is called the *active open*. The other side that receives this packet and sends the SYN segment back performs a *passive open*.

2.4.3 Connection termination protocol

While it takes three steps to establish a connection it takes even four steps to terminate it. The reason is TCP's *half close*. Each sending direction has to be closed independently. This is done by sending a *FIN* segment to the other party. This means that the sender of the packet does not intend to send any more data in this direction. The receiver acknowledges the packet with a sequence

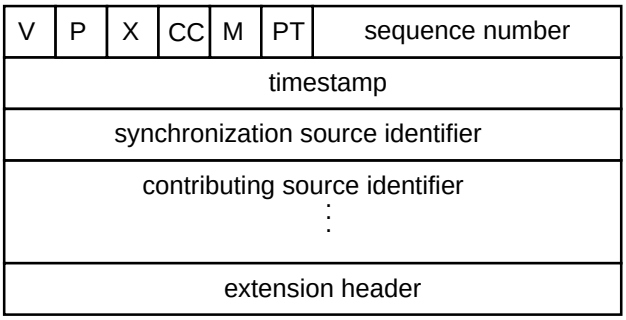

Figure 2.12: Header of a RTP packet

number corresponding to the one sent increased by one. The shutdown of the other direction is not compulsory. Although it is not widely used it is possible to maintain a data flow only in one direction. The common behaviour is that the other party also sends a FIN segment back to the sender to completely terminate the connection. This segment is also acknowledged. It describes the end of the connection and the protocol. The first unit sending a FIN segment performs an *active close*. The other one a *passive close*.

2.5 RTP - Real time transport protocol

This protocol supports end-to-end delivery of data with real-time characteristics, such as video or audio. It is the basis for VoIP telephony protocols. SIP and H.323 uses it to deliver voice data. To fulfill its purpose RTP uses the services of the transport layer protocols. This is in most cases UDP. Since UDP does not have integrated mechanisms to reorder packets or retransmit lost ones, RTP does not have them either. But RTP allows applications to reassemble the RTP packets. The fields enabling it are *sequence number* and *timestamp* in the RTP header.

RTP actually consists of two protocols. The first one is RTP itself. It carries the data with real-time properties. The second one is RTCP, which is a lightweight session control protocol to monitor quality of service (QoS). RTP and RTCP use consecutive transport layer ports when running on UDP [43]. Moreover, RTP defines for each class of application one *profile* and one or more *formats*. A profile is information for the applications to understand the meaning of certain fields of the RTP header. The format describes how the sent data must be interpreted. RTP leaves many protocol details to the specification of the profile and the format. This was an design issue from the beginning.

2.5.1 RTP header

Figure 2.12 shows the header of a RTP packet. The first three lines are always present. *Contributing sources* are only used in special circumstances. The general case will be that an RTP packet carries data coming from one source. The identifier of that source is called synchronization source (SSRC). A special case is e.g. a mixer that merges several RTP streams to one stream for saving bandwidth. In that case the SSRC would be the identifier of the mixer. All sources that contributed to that stream would be listed in the CSRC fields.

The first two bits (*V*) are the version number of the protocol. The current version is 2. The following bit is *P*, the padding bit. It indicates that the payload has been padded, e.g. because of an encryption algorithm that requires inputs of a certain length. *X* is an extension bit that is set if the header is followed by exactly one header extension. Such a header would be defined for a specific header and follow the main header. However, it is rarely used. *CC* is a 4 bit value and specifies the number of contributing sources (CSRC). The marker bit(s) *M* can be used to indicate e.g. the beginning of a talkspurt. Its (Their) meaning is defined in the profile. The following seven bits belong to the field *PT*, which specifies the payload type. Its value defines how the payload shall be interpreted by the application. The *sequence number* is a 16 bit number. It uniquely identifies the source of an RTP stream and allows the identification of packet loss or disordered packets. A random number is used as initial value. For every following packet the sequence number is increased by one. The next field is *timestamp*. It indicates the sampling time of the first byte in the packet's data. To allow jitter and synchronization calculation, this value has to be derived from a linearly and monotonically increasing clock. The frequency of the clock depends on the format of the packet's data. The last field of the mandatory fields in the header is a 32 bit random value uniquely identifying the SSRC. The following zero to 15 fields are filled of unique 32 bit identifiers of sources (CSRCs) that contributed to the packet's payload.

2.5.2 RTCP - RTP control protocol

RTCP is used to send control information to all participants of the session. The information is sent on a periodically basis. A separate connection (pair of ports) is used to deliver the control data. The transport protocol is UDP, as in RTP. RTCP implements four functions: Firstly, it has to deliver information about the quality of the data distribution. It secondly supplies a persistent transport-level identifier for every participant. This functionality permits for instance the indication of the source of the stream. The third function is the possibility of calculating the number of participating parties. This is accomplished by sending the control information to every other participating party. The fourth and last functionality is the support of a minimum session control. RTCP defines several different packet types:

- SR is a sender report and conveys statistical data of an active sender.
- RR is a report of a receiver for statistics of a participant that does not actively send data.
- SDES contains source description items
- BYE indicates the end of a participation
- APP consists of application specific data

The packet format starts with a fixed pattern. It is followed by structured elements of variable length. The packet has an aligned boundary of 32 bits. This allows the concatenation of several RTCP packets to one compound packet. These cumulative packets are sent in one UDP datagram.

Chapter 3

Internet telephony

The Internet is a booming area. With its increasing propagation it has also become more commercial. This introduces new services and usage scenarios. One of them is the transmission of voice and video over the Internet. It is called voice over IP (VoIP). To enable VoIP services it is necessary to send real time data over the network. The protocol for this purpose is RTP, explained in section 2.5. For the establishment of connections between two (or more) parties of a telephone call, a signaling protocol is required. Today there are two main call signaling protocols. One is driven by the telecommunication world, the International Telecommunication Union (ITU-T) (ITU is the former Comité Consultatif International de Télécommunication et Telegraphie (CCITT)) and is called H.323. H.323 is not a single protocol, rather a protocol suite. The other protocol was developed by the Internet Engineering Task Force (IETF) and was named session initiation protocol (SIP). The outline of this chapter are a brief introduction to the functions of call signaling as well as the description of SIP and H.323. Since SIP is not an essential part of this work, the main part of the remainder of this chapter is dedicated to H.323.

3.1 Call signalling

One desired property of IP Telephony is to initiate a phone call. The definition of a call is a multimedia session between several participants. The signaling association between a pair of participants in a call is referred to as a connection. It only exists as a signaling state at the two endpoints. An IP telephony protocol has to accomplish four functions:

- User location
- Session establishment
- Session negotiation
- Call participant management

A user who wishes to establish a call with another user has to find out his location. Otherwise the session establishment request would not reach the partner of the intended conversation. This function is called *user location*. Users can be in different places at different times. They can even be reachable by different means at the same time (e.g. cell phone, computer or a traditional phone).

A further function is *session establishment*. The signaling protocol allows the called party to accept the call, reject it or redirect it to another location. An initiated session can include different multimedia streams (video or audio), different encoding schemes, compression algorithms or even ports and IP addresses.

The third function *session negotiation* allows the negotiation of a set of properties for a session. This is also called a *capability exchange*. During a running session it is possible that other endpoints join or leave existing sessions. This property is administered by the fourth function of the call signaling protocol, *call participant management*.

3.2 SIP - Session initiation protocol

SIP is the call signaling protocol specified by the IETF. It is used to create, modify or terminate multimedia sessions. These sessions can be established with one or more participants. The transport protocol for SIP can be both UDP or TCP.

The design goals of SIP where scalability, component reuse and interoperability. Scalability stands for two things. Firstly, the scalability of the number of sessions. A user can be involved in an arbitrary number of different sessions at the same time. And secondly SIP was designed to work wide-area from the first day [46]. In other words, users can be located far from one another on the network.

The following subsections introduce the basic parts of SIP, the protocol, the components of a SIP environment and finally a simple call flow.

3.2.1 SIP messages

SIP has a similar pattern to the hyper text transfer protocol (HTTP). It is, as well as HTTP, a request-response protocol. SIP moreover borrows much of the semantics and syntax of HTTP, e.g. the textual message formatting, the usage of headers (which is often identical to HTTP) and multipurpose Internet mail extension (MIME) support. Figure 3.1 shows an example message of SIP. The corresponding call architecture is illustrated in figure 3.2. This message would invite the user `Caller` to a session suggesting a phone call using audio stream. The first thing to notice is that the entire message consists of plain text. Secondly it can easily be differentiated between the header part of the message and the body part. The header contains a description of the message type (`INVITE`), the protocol type (`SIP/2.0`), the endpoints of the connection (`From`-field and `To`-field) and the type of the message body. The message body describes the session that the caller wants to establish. There is a particular protocol for this purpose, the session description protocol (SDP). SDP is specified in RFC2327 [18].

SIP methods

The first word in the first line, `INVITE`, specifies the SIP method that is used. SIP methods are the requests that are sent to the other party of a call session. The addressed party answers to the request with a response. The response messages consist as in HTTP out of three digits with an associated textual phrase. For further details about which methods exist or what response codes exist see [35].

```
INVITE sip:called@there.org SIP/2.0
From: Caller <sip:caller@here.org>
To: Called <sip:called@there.org>
Call-ID:12452552@here.org
Content-Type:application/sdp
Content-Length:98

v=0
o=called 123456789
s=A telephone call
m=audio 49150 RTP/AVP 0 3
a=rtpmap: 0 PCMU/8000
a=rtpmap: 3 GSM/8000
```

Figure 3.1: An example SIP message

Addressing

SIP addresses look similar to e-mail addresses. A SIP Uniform Resource Locator (URL) can look for instance like in figure 3.1: `caller@here.org`. The format is always `user@host`. The user part of the address can be chosen arbitrarily. The host part describes the domain the user belongs to. The advantage of this kind of address is that for locating a user the domain name system (DNS) can be used. However, DNS cannot be used if the only given user information is a telephone number. The *IETF ENUM Workgroup* (Telephone Resolution Workgroup) is currently addressing this problem [46].

SDP - Session description protocol

The type of message body as well as its length is described in the header fields of the request message. The corresponding fields are: `Content-Type` and `Content-Length`.

One possible message body type is a multimedia session. The properties of the session are specified using SDP. As illustrated in figure 3.1 SDP uses plain text for the specification of a session. This has mainly three advantages: there are no platform dependent properties to consider such as network byte order, it is easy processable by text-based toolkits such as Perl or Tcl/Tk and this kind of encoding makes it easy to integrate SDP in other application protocols.

Every entry of a session description is composed of the form `type=value`. Thereby `type` always consists of only one letter. The format of `value` is not fixed. A list of all types can be found in [18].

The properties of a multimedia session that SIP covers are:

- Session name and its purpose
- IP address and portnumber
- Start and stop time

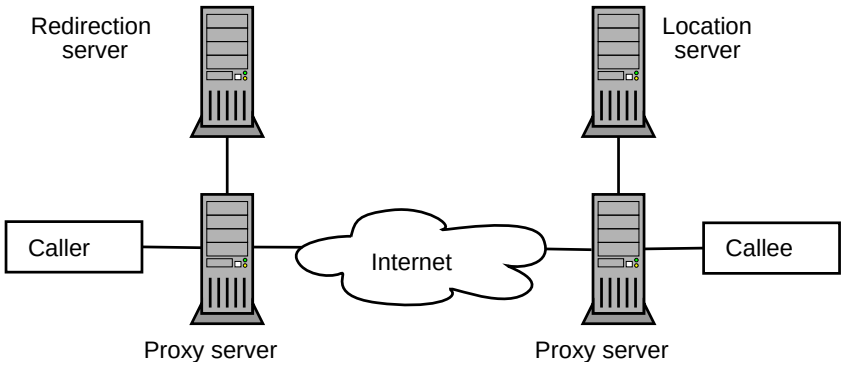

Figure 3.2: A basic call scenario

- Information to receive media
- Information about the bandwidth of the session
- Contact information of the person responsible for the session

The session descriptions are sent by the caller in the body of the *INVITE* request as well as by the called party in the corresponding response. The SDP body describes the capabilities of both parties the caller and the callee [18].

3.2.2 Components of the SIP environment

An important functionality of SIP is user location. A user shall not be bound to exactly one host. To provide user mobility the user must report his actual location to some kind of server. One can see that there are several entities necessary to fulfill this functionality. A SIP environment can therefore contain the following components:

- User agents: They consist of a user agent client that issues requests and a user agent server that responds. In most cases both will be located in one user agent.
- Proxy server: They forward requests to other servers, which can also be proxy servers, or they forward them directly to endpoints.
- Redirect server: They get SIP requests and map the message address to zero or more new addresses. These alternative addresses are returned to the requesting host or server.
- Location server: They map a request's address to the actual host where the user can be reached.
- Registrar: They are servers that accept registration requests of user agents. Registrars are often co-located together with proxy servers or redirect servers. The existence of a registrar is not required. However it can be useful for instance for the mobility of users and for the purpose of user authentication or service subscription. This property can also be used to track users.

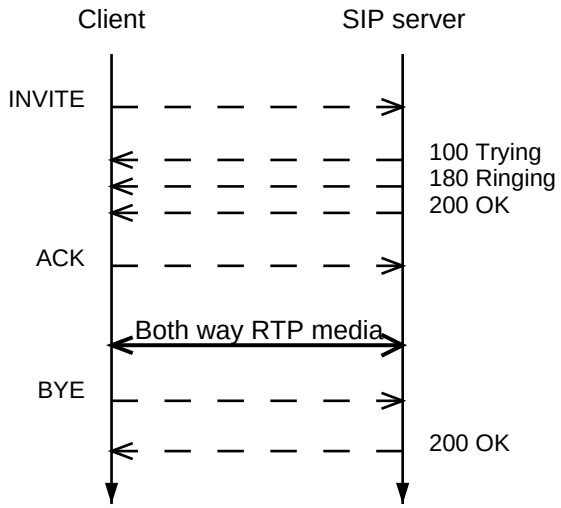

Figure 3.3: Simplified SIP call

3.2.3 Simple call flow

Figure 3.2 shows an architecture for a SIP environment. How the call signaling between two SIP terminals looks like is described in figure 3.3. It shows a simplified SIP call with the most important elements.

A terminal, phone or endpoint in SIP is called a user agent. The call starts at the user agent of the caller. The *INVITE* request is generated and sent to the proxy server. The proxy server contacts the redirection server to deliver all the contact addresses for the target user ordered by priority. Upon having received the contact address the proxy server forwards the INVITE to the callee. The callee can also have a proxy server that tracks the location of the user. A proxy server receiving an INVITE request contacts its location server to resolve the location of the callee. It then contacts the user agent in question which responds to the request [35].

The caller sends the INVITE request. The user agent of the callee responds with the message `100 Trying` indicating that it is processing the call request. The next response is `180 Ringing` which signals, that the user agent tries to reach the user. The callee can either accept or reject the call. The response message for an accepted call is `200 OK`. The caller receives the response and sends an `ACK` message back. With the receipt of that message the callee has also a confirmation about the end of the call initiation. From then on both parties can send their voice traffic to each other. The negotiation of the sessions parameters is done in the `INVITE` respectively the `200 OK` message. For changing the properties of the session either party can send a new INVITE request with the new session description.

The request message and all the corresponding answers build a so called *SIP transaction*. A session is terminated by using the `BYE` request. It is acknowledged by the other party by a `200 OK` message.

3.3 H.323

H.323 is a standard of the ITU-T. It is further a part of the standards for *Audiovisual and Multimedia Systems*, the series *H.32X*. These standards define the transmission of multimedia data over different network types. Their precise assignment's are as follows:

- H.320 [7]: Narrow-band visual telephone systems and terminal equipment
- H.321 [5]: Adaptation of H.320 visual telephone terminals to B-ISDN environments
- H.322 [3]: Visual telephone systems and terminal equipment for local area networks which provide a guaranteed quality of service
- H.323 [8]: Framework and wire-protocol for multiplexed call signalling transport
- H.324 [6]: Terminal for low bit-rate multimedia communication

Recommendation H.323 [8] describes terminals and other entities that provide multimedia services over a packet-based network (PBN), which may not provide a guaranteed Quality of Service. A PBN in this context can be a network based on a.o. local area network (LAN), asynchronous transfer mode (ATM) or fiber distributed data interface (FDDI). The support for audio services is compulsory whereas data and video services are optional. For supplementary services like *Call Transfer* or *Call Hold* there is another series of protocols, *H.450.1 - H.450.12*.

H.323 contains three main types of interactions: The first of them is responsible for all administrative operations of a user. It is called registration admission status (RAS). The second one takes care of call signalling for call establishment and call completion. This is protocol Q.931 [53]. RAS and Q.931 are subsumed to one protocol with the name *H.225.0* [1]. The third type of functions included in H.323 addresses terminal capability negotiation and call control. These are covered in H.245 [4].

This section starts with introducing the components of an H.323 IP telephony system. It is followed by the explanation of the structure of H.323 messages. This is important because later parts, especially the implementation, will list some messages for better understanding. After that the protocols H.225 and H.245 will be described. Finally, the last subsection discusses the different possible call models.

3.3.1 H.323 zones

A zone is the collection of all H.323 entities managed by a single GK. It also can be distributed over several network segments. These segments are connected through routers [8]. There are several different types of entities in an H.323-based IP telephony network. They are classified according to their individual function:

- Endpoint (EP): They are the equivalent to telephones. They can initiate and answer calls. Furthermore they are able to create and to terminate the data streams necessary to communicate.
- Gateways: Its purpose is the connection from an IP Telephony system to the conventional telephony system (public switched telephone network (PSTN)). They translate the signaling from PSTN into H.323.

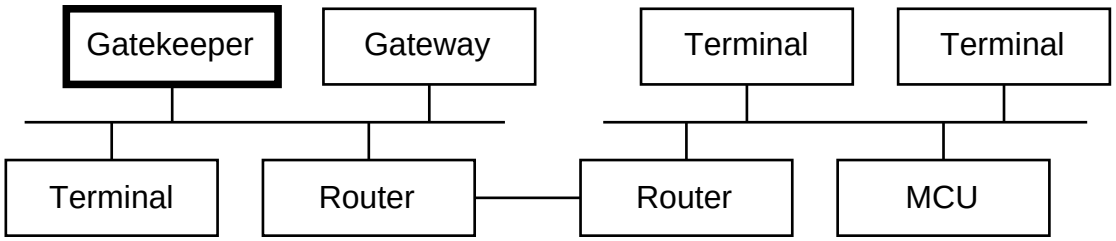

Figure 3.4: A zone in a H.323 Network

- Gatekeeper (GK): A GK serves as an administration unit. It provides five different types of services:
  - Zone management: Each GK is responsible for a single zone.
  - Bandwidth management: It ensures a guaranteed bandwidth for every EP whose call is routed over the GK. Above a certain amount of connections the GK does not allow new ones.
  - Admission control: Every EP in a H.323 network has to register during startup to a GK. This GK allows the EP to setup calls or answer calls from other parties.
  - Call authorization: This service is optional. Call authorization refers to the person initiating the call rather than the endpoint (admission control). Scenarios for such decisions include for instance a *long distance call* or *frozen accounts*.
  - Call management: This feature is also optional. It includes for example status information about active calls, such as the used bandwidth. *Call redirection* also belongs to call management.

- Multipoint Control Unit (MCU): It offers the capability for more than two entities two communicate with one another. It consists of one multipoint controller (MC) and optionally one or more multipoint processors (MP). The MC provides the basic functionality for multipoint conferencing. This includes ad-hoc multipoint conferences as well as capability exchange for all entities involved. Ad-hoc multipoint conferences are calls between two entities that are enlarged to a multipoint conference through the participation of another party. A multipoint processor is responsible for the centralized processing of audio and, or video data of multipoint conferences [8].

- Administrative Domain Back End Service (AD-BES): The AD-BES or just BES is the service interface for EPs and GK. It stores data about all endpoint types and their permissions, services and configuration. There are two ways to implement the BES. The first is that the EP communicates directly with the BES. In this case the messages intended for the BES are forwarded by the GK to the BES. In the second case the EP only interacts with the GK. The GK exchanges messages with the BES to get all the necessary information to respond to the EP's requests.

```
1 Message::= SEQUENCE
2 {
3      userIdentifier     INTEGER,
4      ipAddress          TransportAddress,
5      tokens             SEQUENCE OF Token OPTIONAL,
6 }
```

Figure 3.5: An ASN.1 message example

3.3.2 ASN.1

ASN.1 is used for the representation of the messages of H.323. The abbreviation ASN.1 stands for *abstract syntax notation number 1*. The background to ASN.1 is the idea of providing a notation for defining data structures, a machine-independent encoding for them and tools to produce such encodings from local representations of data in programming languages [33]. It evolved from the CCITT Recommendation X.409 (a part of the X.400 Series). The ISO standards are called *ISO 8824* and *ISO 8825*. ISO 8824 [22] also named ASN.1, contains the notation of the syntax. The encoding rules are specified in ISO 8825 [23] and were called *basic encoding rules (BER)*. There are other encoding rules such as *packet encoding rules (PER), distinguished encoding rules (DER)* or *canonical encoding rules (CER)*. These rules transform the data specified in ASN.1 into a standard format, that can be interpreted by any system applying the same rules.

Overview of the notation

An ASN.1 message is built up of structured types. They can be either primitive types that ASN.1 provides or complex, constructed types. Primitive types are for instance *INTEGER*, *REAL*, *BOOLEAN* or *NULL*. Complex types can be created by using one of the following three constructs: repetitions (*SEQUENCE OF* and *SET OF*), alternatives (*CHOICE*) and lists of fields (*SEQUENCE* and *SET*). Once a type is defined, it can be reused for the definition of other types.

Figure 3.5 depicts an example message. The *message* is defined of a sequence of types. This means that it consists of several different elements. These elements are listed in the brackets, one after the other. An element is listed by its name followed by its type. The separation of elements is done with commas. The elements in the example are: `userIdentifier`, `ipAddress` and `tokens`. The `userIdentifier` is an integer value. Integer is one of the primitive types of ASN.1. The `ipAddress` is of type `TransportAddress`, which is a more complex type. It could consist e.g. out of a sequence of four integer values. The last element of this message is `tokens`, which is a sequence of `Token`. This denotes that there can be more than one `Token` in this message. Because of the keyword *OPTIONAL* one is free to omit `tokens` completely. With this notation it is possible to specify many different and very complex protocols. Some “real world” messages are shown in appendix B.

3.3.3 H.225

H.225 embodies the RAS messages and Q.931. It is defined in ITU-T recommendation H.225 [1]. The RAS messages are the messages concerning the administrative issues of a user. The call signaling part of it is carried out by Q.931. Both protocols use a separate connection for their operation. These are called *RAS Channel* and *Call Signaling Channel*.

RAS messages

The functions contained in these messages concern registration of EPs, admissions to launch calls and status messages. The word RAS is built of the first letters of these functionalities: *R* for registration, *A* for admission and *S* for status. The most common functionalities of RAS are:

- GK discovery: EP searches on startup the GK which it shall register to.
- EP registration: EP registers at its GK.
- Call admission: Before initiating a call, an EP must request permission from the GK.
- Disengage issues: To release a call either EP of a call may request disengagement.
- EP unregistration: Before disconnecting from the GK an EP must unregister.

The RAS messages are sent using UDP as transport protocol. On startup, an EP has to find the GK that it is assigned to. This is necessary because only that GK will allow the EP to establish calls. The corresponding function is called GK discovery. After the successful GK discovery the EP has to register at the GK (EP registration). The registration contains some means of authentication to provide misuse. If the registration process is also finished successfully, the EP can launch calls. But before that it has to request permission from the GK. If the GK has sent a positive reply, the EP is allowed to initiate the call. This is the call admission process. Again, before releasing a call an EP has to contact the GK and send a disengage request. On disconnecting from the GK the EP has to go through the unregistration process.

The communication on the RAS channel is a client-server model. RAS messages are used between EPs and GK. That is to say that a client issues a request and the server responds to it. In most cases, the EP is the client and the GK the server. The answer to a request can be either a *confirm* or a *reject* message. The general term used for both, confirm and reject, is *reply* message. There are always three messages that belong to one functionality.

The specifier of each RAS message consists of three letters. The first letter specifies the type of operation. The letter for GK discovery is a *G*, EP registration has a *R*, call admission an *A*, disengagement a *D* and unregistration an *U*. The last two letters define the type of the message. The possible message types are: request (*RQ*), confirm (*CF*) and reject (*RJ*). Table 3.1 lists RAS messages that are used later in this work. A table of all RAS messages can be found in [1].

Q.931

Q.931 is the ITU-T Recommendation for call signalling of integrated services digital network (ISDN). Call signalling consists of call establishment, call maintaining and call completion.

Q.931 is the ITU-T Recommendation for call signalling of ISDN [53]. Call signalling consists of call establishment, call maintaining and call completion. The establishment of a call, as

RAS message	Explanation
GRQ	Gatekeeper Request
GCF	Gatekeeper Confirm
GRJ	Gatekeeper Reject
RRQ	Registration Request
RCF	Registration Confirm
RRJ	Registration Reject
ARQ	Admission Request
ACF	Admission Confirm
ARJ	Admission Reject
DRQ	Disengage Request
DCF	Disengage Confirm
DRJ	Disengage Reject

Table 3.1: The most common RAS messages

Q.931 Message	Explanation
Setup	initial message to set up a call
CallProceeding	indicates that no more call establishment information will be processed
Alerting	indicates that the user is being informed of the call (ringing)
Connect	finishes the call establishment
ReleaseComplete	initiates the completion of the call

Table 3.2: The most common Q.931 messages

illustrated in figure 3.7, starts with one EP issuing a *setup* message to the desired destination. The destination's EP then issues *call proceeding* to indicate that no more call establishment will be accepted. The destination's EP then informs its user of the attempted call initiation. It indicates this with a ringing signal and sends an *alerting* message back to the initiating EP. The user accepts the call by picking up the receiver. The corresponding signal is a *connect* message. This message ends the call signalling process regarding the call initiation. The next step is the establishment of the multimedia data channels. This is done with the help of protocol H.245 (see section 3.3.4).

Q.931 is transmitted over a reliable channel, such as TCP. The EPs negotiate ports to listen to. The negotiation takes place within the RAS messages (during the call admission phase). Typically the well known port *1720* is chosen.

3.3.4 H.245

H.245 [4] is the control protocol for multimedia communication. It is used to convey end-to-end call control signalling between EPs. This includes capability exchange, master-slave determination and logical channel management. The recommendation has been defined to be independent of the underlying transport mechanism, but is intended to be used with a reliable transport layer (e.g. TCP). The main functionalities of H.245 are:

- Master-slave determination
- Capability exchange
- Logical channel management

Master-slave determination is used to prevent conflicts that may arise when two terminals involved in a call initiate similar events simultaneously. Especially when just one event is desired. To solve such conflicts, one terminal acts as a master and the other one as a slave. The terminals negotiate this roles through applying certain rules. The status of an EP remains the same during the entire call.

The capability exchange procedure ensures that two EPs sending multimedia streams to each other "understand" themselves. Therefore each terminal has to know the capabilities of the other one. If a terminal does not understand a set of capabilities it can ignore them. Every terminal sends its *transmit* capabilities as well as its *receive* capabilities. The capability sets can be reissued at any time.

The logical channel management ensures that a terminal is able to receive and decode data from a logical channel when this one is opened. The *OpenLogicalChannel* message includes a description of the type of data to be transported. Conflicts are avoided through the master-slave determination function of this protocol.

3.3.5 Call flow

From an architectural point of view there are two possibilities of call establishment, *EP-routed call* and *GK-routed call*. In the EP-routed call model an EP directly connects to the desired party. In this case all the traffic is exchanged directly between the two EPs. This includes all protocols of H.323 (H.225 and H.245) and implies that the EP setting up the call exactly knows where the destination resides. This is to say, it has to know the destination's IP address.

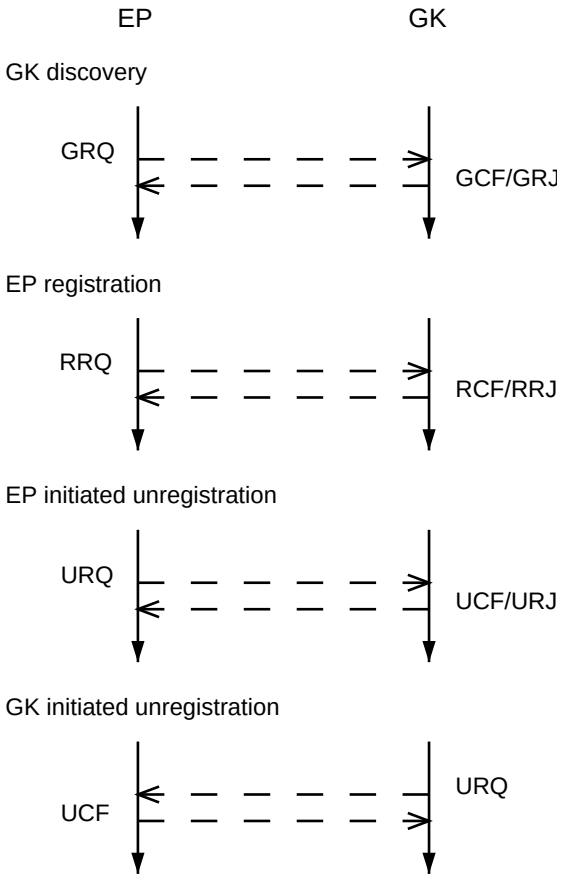

Figure 3.6: Messages involved in the registration process

In the GK-routed call model all calls are established through a GK. This model is scalable and administrable and therefore is used in most business models, e.g. a *subscription based service*. Here, every EP has a *shared secret*. This shared secret is used to prove the identity of the EP's user to the GK. Without this authentication a user is not allowed to establish a call, not even to register. All considerations in the following chapters will refer to this model.

The RAS messages of H.225 are only exchanged between the EPs and the GK. The GK never forwards RAS messages. It routes Q.931 messages and optionally H.245 messages. Media traffic is also never routed through the GK. The EPs directly exchange all the media data belonging to their call. The interactions between the EP and the GK can be divided into two categories: the registration process and the call establishment. Both are briefly discussed below.

Registration

The system is based on the GK-routed model. In this model an EP establishes calls over a GK. This requires a registration procedure. The registration process is composed of two phases. The GK discovery and the registration process. The messages that are involved in the registration process are illustrated in figure 3.6.

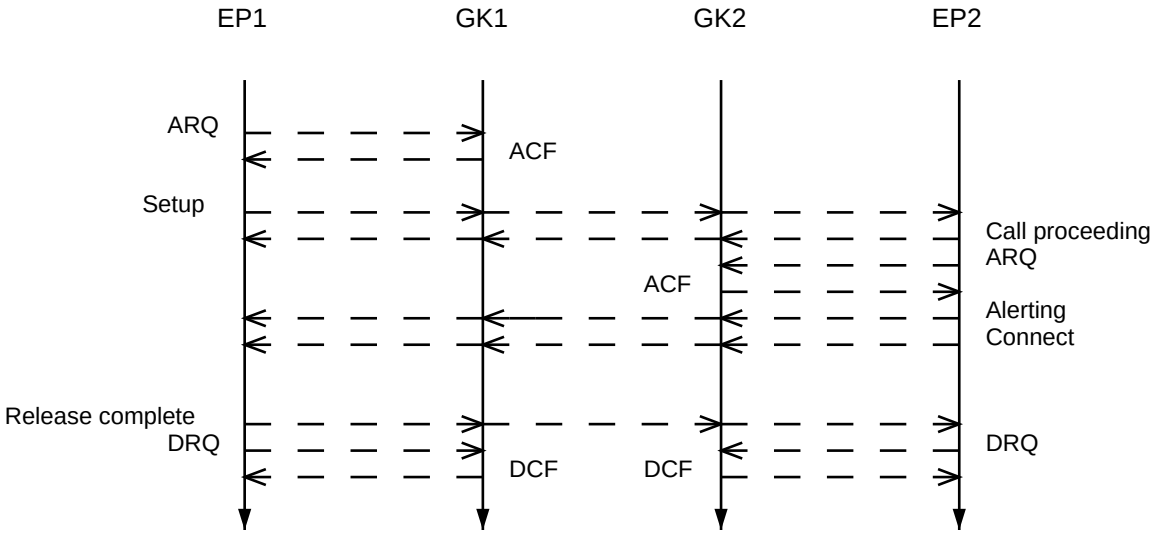

Figure 3.7: Messages involved in the call establishment

Before an EP can start the registration process it has to *discover* its GK. The first message an EP issues is *gatekeeper request (GRQ)*. If the EP is allowed to register with the GK the GK returns a *gatekeeper confirm (GCF)* message. The answer in case of a failure is *gatekeeper reject (GRJ)* (see 3.3.3). A GK discovery is always carried out when an EP is not registered at a GK. This can for instance be either on start up or when a client loses its connection to the GK.

The actual registration is initiated by the EP. The message used for this purpose is *registration request (RRQ)*. The answer of the GK is *registration confirm (RCF)* in case of success and *registration reject (RRJ)* otherwise. To unregister an EP sends an *unregistration request (URQ)*. This message does not have to be answered by the GK. In figure 3.6 the GK sends a response. A different situation is the unregistration by the GK. In this case the GK sends the *unregistration request (URQ)*. Unlike the unregistration of the EP the answering party must send a response.

Call establishment

This procedure includes RAS messages and the Q.931 protocol. Figure 3.7 shows a call setup over two GKs. The first step in a call is to request admission. The corresponding message is the *admission request (ARQ)*. If the GK permits the call by sending an *admission confirm (ACF)* message the real call establishment can take place.

The next step for the EP is to issue a *setup* message that is forwarded to the destination. The destination EP also has to request admission to answer the call. The following response is a *connect* message. This message is the last one of the call establishment of H.225.

After that the terminals have to go through the control protocol H.245 for exchanging the EPs' capabilities and for opening channels for the transport of RTP media.

There are a few variants to this procedure which are discussed in the next paragraphs:

- Pregranted ARQ
- Fast connect
- H.245 tunneling

Pregranted ARQ specifies the possibility of establishing a call without going through the admission process. It is negotiated during the registration process (the RAS messages are: RRQ, RCF, RRJ). It is possible to use pregranted ARQ's to establish calls or answer calls. The client requests the feature in the RRQ message. The GK responds in RCF whether connections using pregranted ARQ will be allowed. It separately grants permission for initiating calls and answering calls. From that point on it is sufficient to send a connect message to establish a call. Especially from a security point of view this feature has to be restricted to as few EPs as possible.

Another variant of the call establishment is *fast connect*. It is the procedure for a setting up a call and negotiating media channels within one *round-trip time*. Generally the round-trip time is the time needed on a network to receive an answer to a sent data packet. In this case it means call establishment in one request-response session. The call control channel does not have to be established, since all the necessary information is handed over during the call establishment process.

In the default call establishment the EPs negotiate a separate channel for H.245. It "carries" the negotiations of H.245. A different possibility to convey H.245 is called *H.245 tunneling*. It is the transmission of H.245 call control messages within Q.931 messages. The advantage of it is that there is no need of opening a separate connection. The call control messages of H.245 are encoded and transported in special fields of Q.931 messages. If there are no pending Q.931 messages, the Q.931 message *facility* is used to deliver the H.245 messages.

Call termination

The first and main message in the call termination process, as figure 3.7 illustrates, is the Q.931 message *release complete*. This message indicates that an ongoing call shall be terminated. An EP issues it and the GK forwards it to the other party of the call. The other terminal does not have to return a response. The second message that an EP sends on call termination is a *disengage request (DRQ)*. The DRQ message is sent to the GK to inform it about the end of the call. The GK sends back a *disengage confirm (DCF)*. Sent by the GK a DRQ forces the EP to drop a call. The EP is not allowed to reject this request. Thus, it may not send a DRJ on a DRQ received from a GK.

Chapter 4

Secure information systems

Every computer system serves a special purpose and stores some kind of data. It is important that only people who have the right to access the system or its data also have the possibility to do it. Abuse of data and damage have to be prevented. Thus, a computer system needs protection. The bigger a system and the higher the number of users is, the higher are the security risks. A special risk is the connection of a system to the Internet. The advantage of a connection with other networks, a high number of users and for business companies a high number of possible customers is paid with the security risk that the Internet imposes.

This chapter gives a general introduction to system security with a focus on network security. It lists different kinds of threats and attacks, as they appear on the Internet. It further describes weaknesses of TCP/IP and introduces in the final section firewalls, a method to protect computer systems that are connected to the Internet.

4.1 Security and dependability

From the point of view of a computer system, security is one part of the ability of maintaining a trustworthy system. This property of a computer system is referred to as dependability [41]. There are four parts that contribute to a trustworthy system.

- Availability: It is a measure for a system to be operational at a certain time. A loss of availability is often referred to as denial of service (DoS). A high availability is often realized through redundant hardware systems.
- Reliability: It defines the probability for a system to perform its functions during a time period. Reliability is different from availability since it is measured over a period of time. It corresponds to the continuity of a service.
- Safety: It indicates whether a system performs its functions correctly or acts in case of a failure in a manner that no catastrophic consequences occur.
- Security: In this case security is meant as a protection of all system resources.

A computer system, dependable to the highest grade, is available all the time (which includes reliability). It does neither crash nor reveil sensitive information. Regarding sensitive data there are two aspects to consider: *confidentiality* and *integrity*. The term confidentiality, as explained

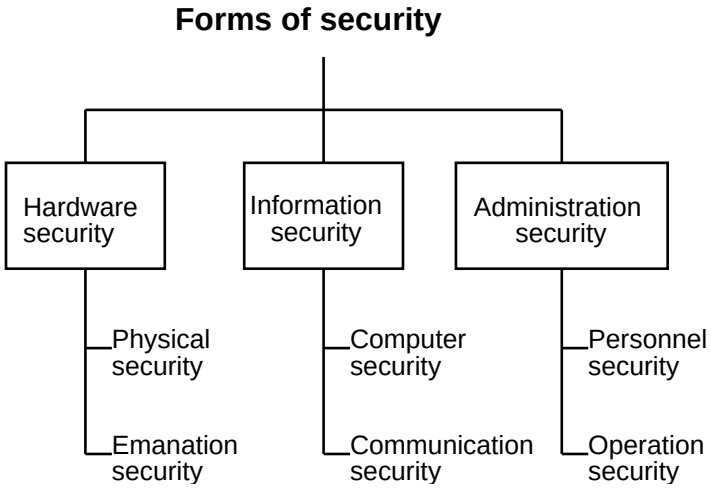

Figure 4.1: Forms of security

in chapter 5, means that data shall not be accessible by unauthorized people. Integrity which is further defined in chapter 5 means that data is not modified in an unauthorized manner during its lifetime. They both impose restrictions for users to create, read, modify or delete data.

Availability , reliability, safety and security are interdependent components. Security protects the system from threats and attacks. It assures an available, reliable and safe system.

4.2 Forms of security

The security of computer system depends on all of its components. Figure 4.1 shows a hierarchical classification. There are three different forms of security:

- Hardware security
- Information security
- Administration security

The first form of security, hardware security, deals with threats and attacks that are related to the system's hardware. It can be separated into two categories: physical security and emanation security. Physical security protects the hardware in the system from external physical threats, such as tampering, theft, earthquakes and water flooding [41]. All the sensitive information in the hardware resources of the system needs protection against all these kinds of threats. For instance an unauthorized person should not be able to remove or add hardware devices. Physical security therefore is responsible for building up and maintaining an environment that is secure enough to contain and handle information. Emanation security protects the system against the emission of signals from the hardware of the system. This contains a.o. emissions from displays (electromagnetic or visible) and audio emissions.

The second form of security is information security. It deals with vulnerabilities in software, hardware and the combination of hardware and software. It can be divided into *computer security* and *communication security*. Computer security covers the protection of objects against exposures

and against attacks making use of vulnerabilities in the system's architecture. It is responsible for a wide variety of problems. These include the access control mechanisms, the mechanisms to enforce the security policy, the hardware mechanisms (e.g. virtual memory), the encryption techniques, etc. Communication security protects transmitted objects. The objects can be local communication or remote communication.

The third and last form of security is administration security. Administration security deals with all the threats that human beings impose to a computer system. These threats can be separated into *personnel security* and *operation security*. Personnel security covers the protection of objects against attacks from authorized users [41]. Every user of a system has privileges to access certain resources. Personnel security contains protection mechanisms against users that try to gain higher privileges or abuse their privileges. It also includes protection against exposures arising from users such as confidential mail sent to the wrong person or a user that forgot to log out. The education of users is very important in this regard. It is actually a mechanism to maintain system security. Another measure is restricting a users access to a system. In general, authorized users impose a higher threat to a computer system than outside attackers. According to statistics only 10% of all computer crimes are performed from outside of a system, whereas 40% are committed by insiders and about 50% by former employees [41]. Operation security enforces the security policy. It deals with the protection of vulnerabilities present in the organization. It defines which actions shall be taken on the occurrence of a security violation and how the system can recover from it. Attackers cannot directly adverse operation security. They can only use a vulnerability to advert computer security or communication security.

4.3 Threats

Threats to a computer system concern all of its parts: hardware, software, data sent or received while interacting with other systems. The adversion of a system can hereby be either accidental, caused by the system itself or an attack. The following three categories are listed in [47]:

- Natural threats
- Accidental or unintended threats
- Deliberate threats or attacks

The first category, natural threats, addresses the possible incidents concerning physical events. These events include all different kind of catastrophes. Catastrophic events can be a threat itself or induce another event that is a threat. The effect of this threats can reach from power outages to loss of data and destruction of devices. The occurrence of these events is not predictable.

The second category, accidental threats, are the result of some improper use or operation. The consequence can be a violation of confidentiality or integrity. They appear either in form of some kind of *exposure* or *modification*. Both hardware and software can expose information. Hardware does this e.g. through emissions, an exposure by software would be a mail sent to the wrong person. Modification can also occur through some resource in an illegal state or an illegal event.

The third category, deliberate threats or attacks, lead to the disclosure of confidential information or the modification of data. Disclosure of information violates the principle of confidentiality and modification of data integrity. These threats are the result of actions taken by people with the intention to steal or modify information or to damage the system.

4.4 Attacks

There are different classifications for attacks. The first one is a differentiation between *direct* and *indirect* attacks. Direct attacks aim directly at a certain object. To access one object, maybe several objects have to be attacked. In an indirect attack, the attacker tries to gather information about an object, without accessing it directly. This kind of attack is a problem for database systems. There it is possible to ask indirect questions to a database and derive confidential information from it.

The second classification is a separation into *passive* and *active* attacks. Passive attacks do not change data. They are carried out by monitoring a system. Data is collected and analysed. An example is traffic analysis. Also encryption does not completely solve that problem because even the fact that there is traffic on the network reveals already information. Another example for a passive attack is the dictionary attack on the password file of a UNIX system. It is explained later, in section 4.4. The other kind of attacks, active attacks, change the behaviour of a system "actively". They can be subdivided into five different types:

- Interception of data
- Interruption
- Creation of data
- Modification of data
- Insertion of data

The most common type for an attack is the interception of data. It is any authorized access to information. One example is the interception of data packets exchanged between two parties. It is also referred to as *eavesdropping*. Interruption includes any delay or disruption of normal operation of a system. In terms of a network connection this means that data does not reach its destination. In case of data creation the adversary sends data created at the attacker's host to the target. The attacker can for instance try to impersonate a valid user or client. Modification of data stands for changing data. The attacker modifies data that passes his host in some way. The attacker's computer is neither the source nor the destination of the data. Data insertion is adding new data to an existing data packet. A special case of it is the appending of data. Here, the original data packet is not altered but send together with a data portion, created by the attacker. All active attacks make use of at least one of these methods. The remainder of this section focuses on and describes possible attacks on the Internet.

Dictionary attack

This attack is a passive one. Its goal is to retrieve passwords for user accounts, for instance on UNIX like computer systems. UNIX with classic configuration uses a file (`/etc/passwd`) to store the names of the user accounts with additional data. This file is readable by all users of the system. The data in this file contains among others the encrypted password and user privileges. The strategy of the attacker is to guess passwords through encrypting dictionary wordlists and comparing them with the entries in the file. According to [30], a survey by Daniel Klein about user accounts in UNIX, about 25% of the passwords could be disclosed with systematic guessing strategies. He used lists of names and dictionary wordlists.

Eavesdropping or snooping

Eavesdropping or snooping is a passive attack where the adversary monitors network traffic and tries to get information about the system. In applications that do not use encryption such as telnet and ftp, the account names and passwords are transmitted in plain text. There an attacker with access to the network obtains the tokens that grants admission to the service. Although it is difficult to get access to the network, encrypted authentication schemes are an essential in the Internet.

Impersonation

Impersonation is the attack where an adversary is pretending to be somebody else to get access to a service or a resource. Depending on the kind of authentication that is implemented, the attacker has to show knowledge of some kind of shared secret.

A particular case of impersonation is IP spoofing: Many applications rely on source address authentication. IP spoofing is the technique where this source address is forged. The attacker has to change the behaviour of his TCP/IP stack to carry out this attack. This requires knowledge about networking and good programming skills.

This technique is also used to overcome firewalls. Here, the adversary replaces his source IP address with the IP address of a host of the inner network. The firewall, the system protecting a network connected to the Internet, interprets the packet as coming from the inside network and forwards it to the destination. This attack and other attacks that exploit TCP/IP are described in more detail in section 4.5.

Denial of service attack

An Internet service is a program running on a host computer awaiting connections from clients. A DoS attack prevents the accessibility of such a service. It is an active and direct attack. The attacker does not aim to steal anything. He simply wants to put the service out of order. But not every time a service is not accessible has to be caused by a DoS attack. This can also be caused by misconfiguration or misuse.

Depending on the nature of the enterprise offering the service this can have rather severe consequences, for instance for an online shop. A DoS attack can be of one of the three following types:

- Physical destruction or alteration of network components
- Destruction or alteration of configuration information
- Consumption of scarce, limited, or non-renewable resources

The modification of a part of the system's hardware architecture presumes access to its location. An attacker can only try to destroy the physical parts of a system through diverting software. A DoS attack on the Internet can therefore only be of type two or three. The alteration of configuration information needs access to the host computer. This implies that the attacker managed to break into the system. The easiest way for a DoS attack is to attack a limited resource like the bandwidth for an Internet service. The impact of such an attack is bigger if a site is attacked from several hosts at the same time. This variant of the DoS attack is called distributed denial of service (DDoS) attack.

Replay attack

The replay attack is an active attack towards a protocol. The attacker typically captures data packets sent from or sent to a host. He eventually modifies them and reuses them to gain access to a certain kind of service. An example according to IP telephony is that the attacker gets a hand on data packets sent from an authorized user that establishes a call and resends them after modifying source and IP address. This can be prevented implementing the two security services *peer entity authentication* and *data integrity*, which are explained in section 5.1.

Buffer overflow attack

This is another very common attack. It is mainly the result of improper software development. This technique takes advantage of the fact that some commands do not check the input data. These applies especially to string processing commands. Through entering more input than a command or program expects it is possible to overwrite system memory. The content of this memory can be e.g. start or return addresses of program subroutines. In the worst case the attacker can inject malicious code that provides him with the privileges of the systems administrator. The countermeasure to avoid "weak" code is to be aware of the flaws contained in operating systems and programming languages.

Man-in-the-middle attack

In the man-in-the-middle attack an attacker manages to break into the connection of two parties. The two parties that participate in that connection think that they communicate with one another. In fact, all the data is routed over the attacker. He has full access to the interchanged data. He can read it, modify it or even send its own data. The fact that the attacker is located in the "middle" of the two communicating parties gives the attack its name. One example for this attack is the establishment of a secure connection by the use of transport layer security (TLS). The weakness of TLS is the session establishment. There, the two parties have to exchange two keys. This key exchange is the possibility for an attacker to get between the two parties. To read more about TLS, see section 6.5.

4.5 Weaknesses of TCP/IP

The attacks explained in the preceding sections can be applied to all layers of the network architecture, also to the TCP/IP stack. As [51] states:

> "TCP/IP and most of its protocols and utilities were not written with security as priority. They were designed for functionality and portability."

The TCP/IP stack does have some weaknesses as well as its implementation may have. Some exploits are explained in the following paragraphs and concern all implementations. One vulnerability that is not restricted to the TCP/IP stack and was already explained in section 4.4 is the buffer-overflow vulnerability. Buffer-overflow vulnerabilities should not exist but unfortunately may appear in individual implementations of the TCP/IP stack.

The points listed above are ordered according to the layer they appear. IP fragmentation and source routing belong to the IP layer, as well as ICMP attacks. SYN flooding, UDP spoofing and the TCP sequence number attack are carried out on the transport layer.

IP fragmentation

IP fragmentation is the property of IP to split up packets that are too large for the underlying layer. For instance IEEE 802.3 [30], the Ethernet standard, allows a maximum payload length of 1492 bytes. Each packet gets its IP header with an identification number and an offset value of the data in the IP packet. The packets are then sent to the destination and reassembled to form the original IP datagram. In case of a fragmented TCP packet, the header information is only available in the first datagram. This is a problem for firewalls that should throw away fragmented packets. However, modified TCP implementations could evaluate also fragmented packets and obviate the firewall system [30].

Source routing

IP is an unreliable connectionless protocol [51]. The only information about the destination on the IP layer is the IP address. Routing decisions are made according to this address. Source routing is an option of IP to specify the path of hosts that an IP packet has to take. There are two different kinds of source routing. The first one is *strict* source routing. Strict source routing defines all hosts that a packet has to pass. The hosts even must be traversed in the right order. Otherwise an ICMP error message is generated. The second possibility is *loose source routing*. Here, only some hops of the path are given. This kind of routing is dangerous because this option can be used together with IP spoofing for an attack. An attacker, on its host A (attacker), intends to attack a system D (destination). He creates all packets with an IP source address of a not existing system S (source). Additionally, the attacker sets the option loose source routing and adds its host A. All data seems to be exchanged between S and D. However, because lots of protocols use the same route in the reverse direction, the attacker receives all the traffic. As a defense against this attack, firewalls should neither allow source routing nor packets that arrive on the Internet network interface but have a source IP address that belongs to the internal network.

Attacks based on ICMP messages

ICMP provides status information and control capabilities to the IP protocol. Some attacks are based on these messages. These attacks are DoS attacks. They shall deny the accessibility of hosts or even crash them. Three messages are used in particular for these attacks: the *echo request*, the *redirect* message and the *source quench* message.

An attack based on the first ICMP message, the echo request, is the so called *ping of death*. The ping of death is a DoS attack. Echo requests normally have got a size of about 64 bytes. The ping of death sends echo requests with a size larger then the maximum number or larger than 65536 bytes. This attack proved harmful to a lot of operating systems [51].

Redirect messages are the second type of messages. They were introduced to IP to be able to modify packet routes in case of the discovery of a faster routing path. Unfortunately there are no means of authentication for these messages. Therefore they can be used to cause DoS attacks or redirect traffic over hosts controlled by the adversary. Then, the attacker can perform further malicious acts [51].

The third type of messages, source quench messages, are sent to indicate that the system is overloaded. On a received source quench message a sender reduces the transfer rate until no source quench messages are received. The misuse of forged source quench messages can therefore be used to perturb traffic.

UDP spoofing

UDP is a connectionless protocol. UDP datagrams are sent from one host to another. There is no correlation between sent datagrams. It is not possible to detect whether subsequent datagrams belong together. UDP does not add any mechanism with sequence numbers or confirmation to IP. The only mechanism to "verify" packets in UDP is to check its IP address. Thus, it is possible for an attacker to insert forged datagrams using IP spoofing. Because of these flaws systems with sensible data or services should use TCP as transmission protocol.

SYN flooding

TCP is a session-oriented protocol. To start a TCP session the so called three-way handshake protocol has to be carried out (as described in section 2.4). The SYN-flooding attack takes advantage of this mechanism. SYN-flooding is a DoS attack. The attacker sends requests to open a session with spoofed an IP address that does not exist or is not accessible at the moment of the attack. The host responds with ACK-datagrams and waits for datagrams containing a set ACK-flag. But it never gets them because the IP address does not exist. Eventually the connection attempt times out and releases the memory that it used. However, the attacker keeps on sending connection requests until the service runs out of memory and denies further connection requests.

An important detail regarding this attack is that the spoofed IP address cannot be reached at the moment of the attack. Otherwise the host with the spoofed IP address would send datagrams telling the attacked host to terminate the connection. This contradicts the intention of the attacker to keep the target busy.

TCP sequence number attack

This technique allows an attacker to overcome a security system whose access control mechanisms are based on IP addresses. The prerequisite for this attack to succeed is a successful injection of IP spoofed packets into the local network of the target system.

TCP offers a connection oriented service. A TCP connection appears to be a stream to the participants. The proceeding of the session is indicated by sequence numbers. The sequence numbers of consecutive packets must exactly match with the amount of sent data. If an attacker is able to predict the sequence number, he is capable of inserting data packets into that stream (with IP spoofing).

The adversary starts the attack by examining the behaviour of the sequence-number generator of the system. He does this through requesting connections to harmless services. If the attacker finds out how the sequence number generator works and manages to predict sequence numbers, he can start the actual attack. He issues a request with a forged IP address to a critical TCP port on the target host. The server responds to the request and sends data to the system with the forged IP address. At that time the computer with the forged IP address must be disabled, for instance with a DoS attack. Otherwise it would respond to the server that it did not send that data packet and request the termination of the connection. If the system is disabled, the attacker can send messages

to the server. He can proceed without seeing the actual response. This is possible because the attacker knows the protocol it attacks and therefore the data that the response of the server must contain. More important is, that the attacker is able to guess the right sequence numbers of the protocol. If the access management of the server is based only on IP addresses, the attacker can establish a connection with it. He may perform whatever command he want. The attack succeeds. However, he does not get any responses from the subverted system, since this one sends them to the forged IP address.

4.6 Firewalls

Because of the dangers and risks in the Internet covered in the last section a company should not connect their network without any protection to the Internet. Such a protection is given by a *firewall*. Generally a firewall is located where a network is connected to the Internet. On a point where all data sent to other networks has to pass. These points are the first goals for an attacker. Therefore, special concern has to be put on their configuration.

A firewall performs checks according configured criteria and allows only explicitly permitted traffic to pass. For instance, it may check the validity of IP addresses, packet headers, as well as the format of protocols. The last point, checking the format of protocols, is particularly effective. Some services are bound to well known ports and use a certain, known protocol. One example for such a protocol is HTTP. HTTP servers typically use port 80 for its operation. A client requesting a connection to this service has to follow the protocol to get access granted. As a consequence a firewall can determine to a certain grade what information a connection request must contain. These controls assure the properness of the protocol. However, they are very time intensive and reduce the speed of a connection. Traffic routed over the firewall can also be logged to be analysed and checked for possible intruders or attacks.

Only data packets passing a firewall can be checked by it. A user connecting to the Internet e.g. via a modem that is placed behind the firewall (behind means inside the network that the firewall shall protect) surpasses it and therefore surpasses also its protection mechanisms.

Security and firewalls in general are not very convenient. This is due to the restrictions that security imposes to possible user actions. It is therefore very important for a company to conceive its employees of the necessity of the undertaken security measures. Security trainings are needed. The social and human aspect plays a very important role in security and may not be underestimated. The remainder of this section will after a short and general introduction describe different kinds of architectures and firewall techniques.

4.6.1 Security strategies

Achieving a secure company network is a very complicated task. The first thing to do is to define the system that shall be secured. Who or what is part of it? What is or shall be done on the network? What kind of traffic shall be allowed to get out into the Internet and what is allowed to get in? As soon as the components of the system are known one can start to develop security strategies. [51] lists four strategies for the design of a firewall:

- Restriction of user rights to the highest possible extent
- More than one security layer

- Low revelation of information about your system
- Security through obscurity

A basic principle of a firewall is the restriction of user rights as much as possible. Only people who need to access a certain service shall be allowed to do it. It reduces possible points of attacks and reduces the damage done in case of a subversion. Special concern has to be put on accounts with superuser or administrator rights. Only staff that is in charge of a system shall have superuser privileges.

Another general rule is to implement more than one security layer. One possible measure for this rule is a system structured in layers. Each layer gets different security mechanisms assigned to it. An example is a network that is divided into two subnets. One network conveys data with higher sensibility than the other one. A way of protecting these networks is the utilization of a packet filter. A packet filter is a computer or router that lets only data pass that fulfill certain conditions (see section 4.6.2). The network with the confidential data may be protected by an additional packet filter with stricter rules. A second way for the implementation of more than one security mechanisms are redundant security methods. An example for this case is the use of two packet filters for the same traffic. Here only packets that fulfill the criteria of both of the filters can pass. In this way the failure of one security method does not unveil the entire system. The damage can be reduced or even prevented.

Before an attacker can launch a successful attack he has to find a valuable target. He needs information about the network, he wants to attack. Because of that as few information as possible shall be revealed about the system. Some web browsers for example send information about the browser or the operating system in simple HTTP-requests.

Another security strategy, that is also very common, is security through obscurity. It is based on the fact that a vulnerability that is not known can not be exploited. However, it shall never be applied as the only strategy. Basically the less an attacker knows about a system the longer it will take him to break its security. In practice, security through obscurity is not a good solution. Too often in history this kind of security was meant to be secure and has been broken [51].

4.6.2 Firewall techniques

Bastion host

The bastion host is a the computer that is put on the border of a network to the Internet. Because of its location in the system, it is the first target for an attacker. Therefore it has to be specially secured. A bastion host runs services that are accessible from the Internet or application proxy server (explained later in this section) that allow connections from the LAN to the Internet.

Criteria in the choice of the host include: the number of services to run, user accounts, operating system, CPU power, hardware configuration, the actual place where the bastion host shall be and the place in the network architecture. It is also important to consider the case of a successful attack. Everything that is not necessary for the operation of the bastion host must be turned off. Otherwise an attacker may be able to use it to further explore or damage the internal network.

Packet filtering

Packet filtering is a mechanism that decides whether packets are allowed to pass. These decisions are done by packet filtering rules. Packet filter typically run on devices that connect networks.

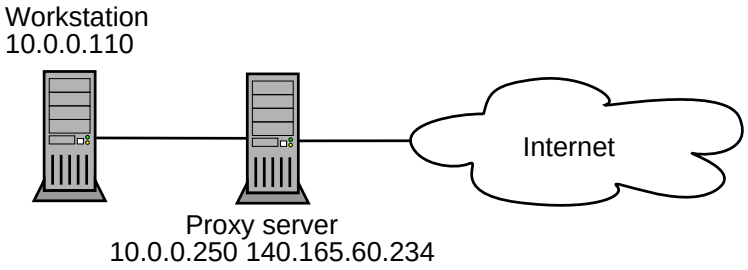

Figure 4.2: Schematic architecture of a connection including a proxy server

Such appliances are called *router*. A router running a packet filter is also called a *screening router* [51]. Screening routers were the first firewalls. Nowadays they are only one part of it. Packet filters are used on the border of every network to the Internet (see 4.6.3).

The rules of the packet filter depend on the requirements of the firewall. Examples are the number of offered services, whether connections from internal hosts to the Internet shall be restricted or not and if there are trusted servers on the Internet. The book “Practical Firewalls” [51] lists the following items as information for the creation of filtering rules:

- Network interface
- Source and destination IP address
- Options of the IP protocol
- The message type of ICMP messages
- The *ACK*-bit for TCP packets

Each data packet is sent from a source to a destination. As a consequence, a filtering rule consists of a source part and a destination part. Each part specifies a network interface, an IP address or an address range and a port number. The specification of the network interface is very important. LANs have usually well known IP address ranges. In each of the first three classes of IP addresses (see figure 2.6) one network address is reserved for private networks. These IP addresses are not used in the Internet. Therefore, packets with a source IP address of a private IP address space should only arrive on the network interface of the LAN.

Further things a packet filter can check are certain options of IP, such as source routing. Source routing can be abused for attacks as section 4.5 illustrates. Other possible sources for attacks are special types of ICMP messages. These attacks are also covered in the section 4.5. Packet filters should not allow these options and messages, to avoid their abuse.

Application proxy server

An application proxy server, also called proxy server, is a program that acts on behalf of the client. It is a server program that works on the application level and forks incoming requests for clients to servers on the Internet. The proxy server runs either on a bastion host or on a dual-homed host (see section 4.6.3). From the Internet it looks like the source of all requests is the computer that runs the proxy server. However there may be many clients using the service of the proxy. The reason

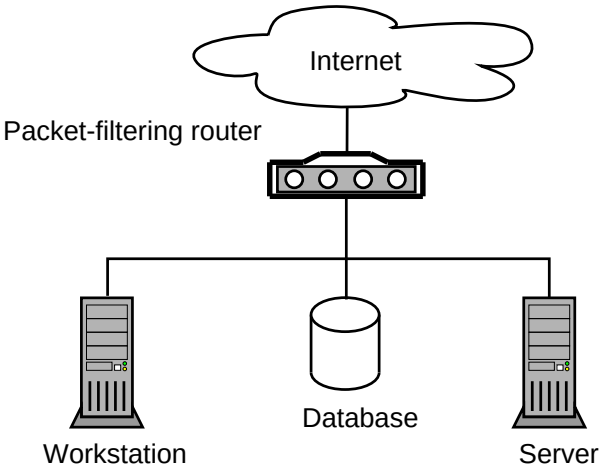

Figure 4.3: LAN protected by a packet-filtering router

for this behaviour is network address translation (NAT). The source IP address of the request is exchanged with the IP address of the proxy server's host. Responses have set the host's IP address as destination IP address. The proxy server maps the response to the right client, inserts his IP address as the destination IP address and forwards the packet to it. Because the proxy is the only host that communicates with the Internet it also is the only one that needs a real IP address.

One main difference between a proxy server and a packet filter is that the packet filter acts on the IP layer or transport layer and the proxy server is on the application layer. A second difference is that the proxy server must "understand" the application. The book practical firewalls [51] states:

> "A packet server can be programmed to pass or drop network packets based on a limited amount of information found in the packets header. The proxy server is application-specific and can be programmed to allow or deny access to a service based on the functions the user wants to perform."

Figure 4.2 shows a connection from a client to the Internet. The workstation that issues the request is located on a LAN. This is indicated by the IP address 10.0.0.110, which is an IP address for a private network. The proxy server runs on a dual-homed host. This can be seen on the two interfaces (IP addresses) it has got. A connection to the Internet works as follows: The workstation issues a request to the proxy server. The proxy server performs some checks on the request. The rules for the checks must be configured by the administrator. They test whether the request fulfills the proxy server's application protocol. If all checks were successful, the proxy server exchanges the source IP address with its own one and sends the request to the server in the Internet. This server has to assume that the actual client is the proxy server. On receiving responses from the Internet, the proxy server checks the protocol, maps the response to the right client and forwards the message.

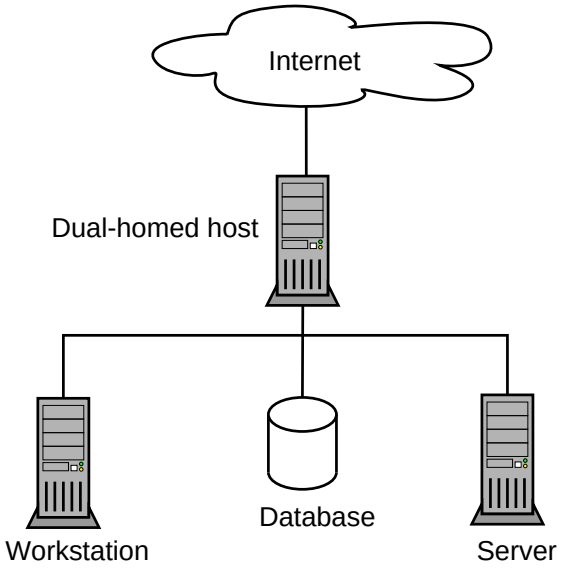

Figure 4.4: Dual-homed host architecture

4.6.3 Firewall architectures

Single-box architecture

The network architecture is shown in figure 4.3. It consists of a LAN that is connected to the Internet by only one single point. This point usually is a router. All the traffic to or from the Internet has to pass this machine. Therefore the firewall, a packet filter, is set up on the router.

The advantage of this architecture is its simplicity. It is easier to operate than other systems with several security layers. As a consequence it is also cheaper than those systems. A disadvantage is that the single-box architecture is not very flexible and only suitable for small networks with simple requirements. Another disadvantage is that in case of a compromise of the firewall the inner network is left without protection. This technique should therefore only be used if computers on the network possess already a high degree of host security and the number of used protocols is restricted [54].

Dual-homed host

In this case one single host takes the functionality of beeing the connection point to the Internet. A dual-homed host is a computer that has two network interfaces. One is connected to the internal network (LAN), the other one to the Internet. The routing of packets from one network interface to the other one is disabled per default. The result is the absence of a direct connection between the two networks. This connection is realised via proxy servers. In this way one can control what traffic is allowed to reach the Internet and vice versa.

The advantage of a dual-homed host is the high degree of control over the traffic to and from the Internet. If all proxy servers are switched off there should be no packets with an origin outside of the network. Otherwise there is a security problem [54]. The configuration of a dual-homed host is critical. An attacker who manages to overcome the dual-homed host has got access to the entire network. A DoS attack that leads to a crash of this host cuts the access to the Internet.

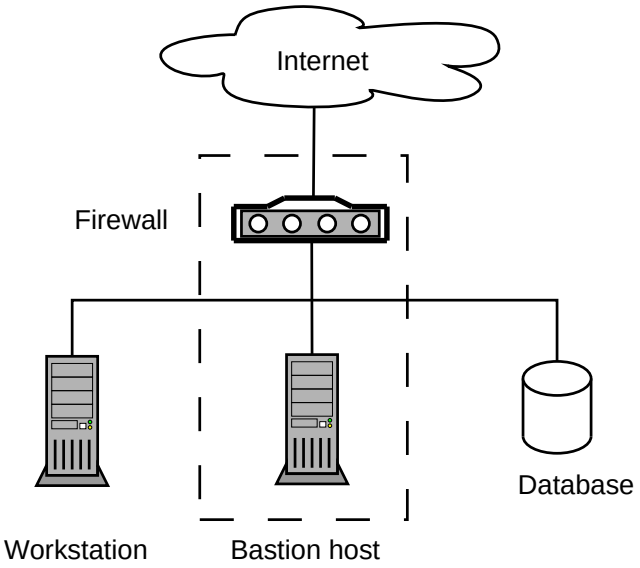

Figure 4.5: Architecture with a screened host

This is particularly critical for a company that is dependent on Internet access. According to [54] dual-homed hosts are suitable for networks that have the following properties: the traffic to the Internet is low, the Internet connection is not absolutely necessary for the companies business, the inner network does not host any services for the Internet and does not contain sensible data.

Screened host

The security in this model is provided by a packet filtering router in combination with a bastion host. Figure 4.5 shows this in the dashed box. Here, the whole local network is connected via the router to the Internet. However the packet filter is configured in a way that only the bastion host is allowed to set up connections to the Internet or accept requests from there. The bastion host has to be configured in a secure way with conscious to its critical task. The connectivity for workstations from the internal network is provided by application proxy server.

The security of this architecture is as good as the security of the packet filtering router. Once the router is compromised, the inner network has no protection. Another disadvantage is that if the bastion host does not hold an attack there is no security left between the host and the rest of the network. It is therefore not well suited for a web server. To maintain a minimal level of protection, in case of a successful attack, the other hosts in the network should also have a high degree of security.

Screened subnet

In all the architectures mentioned until now there was only one “barrier” between the Internet and the LAN. If this one is successfully attacked the security of the hosts depend on each host itself. This is different in the screened-subnet architecture. Here the network is splitted in two areas. In the first zone reside the application proxy servers and the computers that host services. In other words everything that interacts with the Internet is located here. This subnet is separated

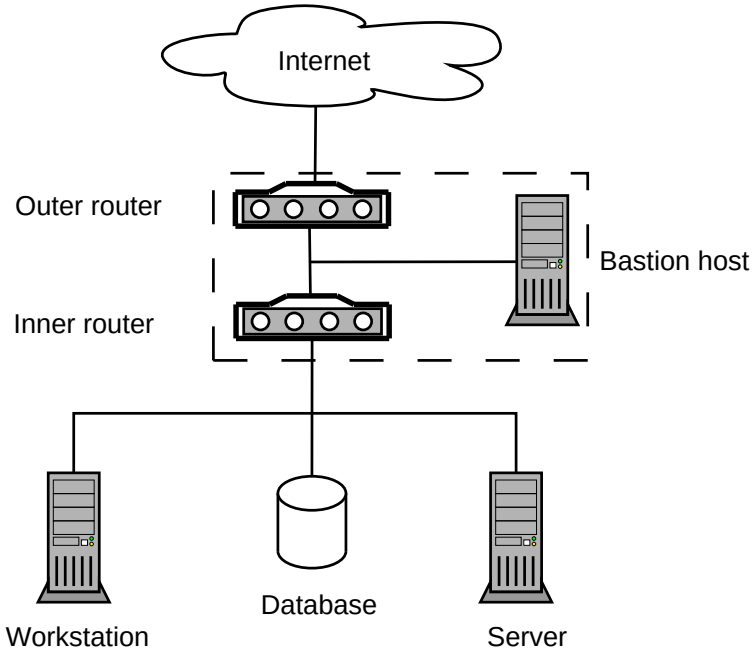

Figure 4.6: Architecture with a screened subnet

by a packet filtering router from the Internet. The router is configured in a way that only traffic dedicated to one of these machines is allowed to pass.

The second zone or subnet is the LAN where computers exchange confident data. This subnet is separated by a packet filtering router from the intermediate network. It lets only data pass that is destined for network elements in the screened subnet, the first zone. Figure 4.6 shows a simple example for such an architecture. In the figure the computer in the intermediate network is a bastion host. The packet filtering router on the boarder to the Internet is called the “outer router”, the other one “inner router. An attacker who successfully compromises the outer router or the bastion host is still not able to access all the data inside the inner network. He still has to overcome the inner router.

Such a network on the boundary between the Internet and the actual network is often called a demilitarized zone (DMZ). A DMZ of course can contain more than one host. Of course there are more complicated configurations. But they are mainly combinations and variations of the mentioned architectures.

Chapter 5

Cryptography

After generally talking about the security of an information system, one has to develop an understanding for algorithms and mechanisms that allow the securization of data. The first thing to do is to define the objectives of information security. Regardless of who is involved, all parties of a transaction must have confidence that the objectives of information security are met [9]. ISO/IEC 7498-2 [25] calls these objectives *security services*. There are five different categories of security services:

- Identification and authentication: This function provides the identification of both entities and data origin. They are called *peer entity authentication* and *data origin authentication*.
- Authorization: It grants access to certain resources or to entities found to be "legal".
- Confidentiality: It is a service that conceals information from anybody who is not authorized to have it. It is also referred to as *secrecy* or *privacy*.
- Integrity: It ensures that data is not altered by unauthorized entities. It is also called data integrity.
- Non-denial/non-repudiation: This objective prevents an entity from denying previous commitments. It offers a possibility to trace whether an entity performed an action or not.

Cryptography provides the technical means to implement these services. To be able to apply cryptographic methods it is important to know what purpose they fulfill, their functionality and properties.

This chapter starts with describing the most important security service for the analysis in chapter 6 and the implementation in chapter 7: authentication. It afterwards gives an overview about basic and widely used cryptographic primitives. These are *hash functions*, *symmetric key cryptography* and *asymmetric key cryptography*.

5.1 Authentication

This section is about a basic need in the electronic world of today. The need to be sure that the opponent we deal with really is who he claims to be. Furthermore, nobody shall be able to impersonate other parties. Names for these techniques are among others *identification*, *entity authentication* and *identity verification*.

Identification mechanism have several objectives:

- Successful authentication
- Non-reusability of exchanged authentication data
- Impersonation protection
- Resistance against observation

Successful authentication can be described as a protocol carried out by two parties A and B, where A authenticates at B and B at the end accepts the identity of A. Reusability means the reuse of information obtained during the authentication to impersonate one party at a third party. In this example a reuse of information would be if B could use the data from the performed identification protocol to authenticate at a third party C as A. An identification mechanism shall also avoid impersonation. In the example used here, this would be an impersonation of A to B. In other words, a third party C may not be able to authenticate as A to B. These properties must remain true, even if the adversary is able to observe a large number of authentications between A and B.

Properties of identification mechanisms

The security of authentication can be build on *something known* like a password or a *PIN number*. It can also be build on *something possessed*, like a chipcard or on *something inherent*, for instance a physical characteristic like biometrics. An application built on such a technology is, for example, the access control to a resource. The identification mechanism could be a password scheme. Password schemes are very common methods for access control. One example is the UNIX-password scheme. Typically, password schemes are weak authentication methods. In general identification mechanisms have three properties [9]:

- Identification or entity authentication: It assures one party (through a validated evidence) of both the identity of a second party and the actual participation of the same party at the communication. Here, only necessary data is transmitted. Entity authentication requires a confirmation of authentication through an identifier obtained in the actual communication.
- Mutual and unilateral authentication: In the first case both parties authenticate to each other whereas in the second case only one party identifies itself.
- Data origin authentication: It provides to one party that receives a message the assurance of the identity of the party that originated the message [9]. This method typically does not provide timeliness guarantee. The receiver does not have to be in an active communication with the sender. Data origin authentication provides prove of the knowledge of the key. It includes by definition also data integrity [9].

Other terms that concern authentication are: *message authentication* and *transaction authentication*. Message authentication is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source (and data integrity, but no uniqueness and timeliness guarantees). Transaction authentication provides message authentication together with uniqueness and timeliness guarantees. The last two properties are typically obtained through time-variant parameters. These include random numbers, timestamps and challenge-response protocols [9].

Another security service that is often provided together with authentication is data integrity. It assures to a second party that data did not change on the way from its sender to the receiver. The reason that data origin authentication and data integrity often are provided at the same time lies in the cryptographic mechanisms used.

5.2 Hash functions

Hash functions play an important role in modern cryptography. In the book "Applied Cryptography" [9] a hash function is defined as: a computationally efficient function mapping of binary strings of arbitrary length to binary strings of fixed lengths. They have three basic properties:

- Preimage resistance: It means that it is computationally infeasible to find an input which hashes to a certain output.
- 2^{nd}-preimage resistance: It is computationally infeasible to find any second inputs which have the same output as the specified input.
- Collision resistance: It is computationally infeasible to find two distinct inputs which hash to the same value.

The first property *preimage resistance* means that it is not possible with considerable effort to retrieve the input of a hash function by only knowing the output. The probability for a hash function with a n-bit hash value to get mapped to a particular n-bit hash value is $1/2^n$. This is important because it can be used to prove whether some data was changed or not. Thus, hash functions can be used to provide data integrity. A typical usage of a hash function is to apply it to a message and send the message together with the output of the hash function applied to the message to the destination. The receiver can prove by applying the hash function to the message, whether the messages was modified during the transmission or not. The second property 2^{nd}-*preimage resistance* ensures that it is not feasible to search a second input to a hash function to get a certain output. Hash functions that fulfill these two properties are also called one way hash function (OWHF). The third property denotes that there is a low probability to find two inputs that result in the same output. A hash function that provides 2^{nd}-preimage resistance and collision resistance is called collision resistant hash function (CRHF). Depending on the type of a hash function and of the way it is applied it can provide integrity or authentication or both. The two different types are: *unkeyed* hash functions and *keyed* hash functions. An unkeyed hash functions is also called modification detection code (MDC), a keyed hash function message authentication code (MAC).

5.2.1 MDC - Modification detection code

MDCs are unkeyed hash functions. The security service they provide is data integrity. Thus, they are used to verify the correctness of data. Unkeyed hash functions can be categorized based on the nature of the operations of their internal compression functions. The three categories of iterated hash functions studied most to date are hash functions based on *block ciphers*, customized hash functions and hash functions based on modular arithmetic. A block cipher is an encryption algorithm that encodes a block of data into a block of the same length. The idea of the first type of MDCs, the ones that are based on a block cipher, is that if an efficient implementation of a

block cipher is already available, it can be used as the central component for the hash function. Examples for this type of hash functions are: *Davies-Meyer hash*, *MDC-2* and *MDC-4*.

The second type of MDCs, customized hash functions, are specifically designed for hashing. They are developed with speed and independence from other system subcomponents in mind [9]. The hash functions of this type that reveived the greatest attention in practice are based on the MD4 hash function. MD4 is specified in RFC 1320 [36]. It was designed specifically for software implementation on 32-bit machines. Hash functions belonging to the MD4 family are:

- MD4: It is a 128-bit hash function. Its design goal was that breaking it should require roughly brute-force effort. It is now known that it fails to fulfill this goal.
- MD5: Security concerns regarding MD4 motivated the design of MD5. MD5 is very popular now in practice. This algorithm also has now been found to have weaknesses. It is defined in RFC 1321 [37].
- RIPEMD-160 [45]: It is also based on MD4 and knowledge gained through the analysis of MD4, MD5 and RIPEMD was taken into account during the development.
- SHA-1: It was proposed by the U.S.National Institute for Standards and Technology. The number of the standard is FIPS 180-1 [38]. SHA-1's hash value is 160 bit. SHA-1 and RIPEMD-160 appear to be of comparable strength today.

The third type of MDCs are based on modular arithmetic. Here, the basic idea is to construct an iterated hash function using mod M arithmetic as the basis of a compression function. Two motivating factors are the reuse of existing software or hardware for modular arithmetic and scalability to match required security levels. Significant disadvantages of this type of MDCs include speed and a history of unsecure proposals.

5.2.2 MAC - Message authentication code

MACs provide a way to check the integrity of information transmitted over or stored in an unreliable medium. They also are referred to as *keyed hash functions*. Typically, message authentication codes are used between two parties, that share a secret key, in order to validate information transmitted between these parties. The sender appends an *authentication tag* to a block of data. The authentication tag is computed as a function of the data and the shared key. At reception the receiver recomputes the authentication tag with the received data and the shared key, and accepts the message as valid, if the result matches the tag attached to the received data [12]. Because MACs use shared secret keys to compute the hash value, they provide data origin authentication or message authentication. Another security service is the data integrity of the hashed data.

The most commonly used MAC algorithm is based on a block cipher. It uses cipher block chaining (CBC). Its name is *CBC-MAC*. MAC's can also be constructed from MDC's. A further approach is to design MACs for specific purposes. Especially designed MACs are called *customized* MACs (e.g. message authenticator algorithm (MAA) [20] or MD5-MAC [11]).

HMAC

HMAC is a MAC mechanism. It can be used with any iterative cryptographic hash functions, for instance with MD5 or SHA-1. The calculation process and the verification process require

also a secret key. The cryptographic strength of HMAC depends on the properties of the underlying hash function [29]. The construction of HMAC was done with special concern about the reuse of available hash functions, especially those who perform well in software. It preserves the original performance of the hash function without a significant degradation. An example for this mechanism is *HMAC-SHA1-96*. It is used in H.235-Annex D [2] as standard for securing H.323. The supported security service is peer entity authentication, data origin authentication and data integrity. HMAC-SHA1-96 is a HMAC algorithm using the hash algorithm SHA-1 as underlying hash function. The number, 96 at the end of its name, describes the number of bits taken from the resulting output to represent the authenticator thus reducing the predictability of the hash functions output (preimage resistance).

5.3 Symmetric key cryptography

The main property of symmetric key cryptography is that for encryption and decryption the same key is utilized. The two participating parties have to agree upon the key before the data is encrypted and exchanged. This can be reached for example through a protocol that allows the establishment of a key on an unsecure channel. Once, both parties know the key, the sender can begin to encrypt messages and send them to the destination. The receiver is capable of decrypting the messages because it also knows the key. A secure cipher is characterized by the fact that all the security lies in the key, not in the algorithm. In that case the knowledge of the algorithm is insignificant. It is important that the key is hard to guess. Therefore the key space needs to be accordingly large. Examples for symmetric key ciphers are *3DES*, *Blowfish*, and *IDEA*.

5.3.1 Stream cipher

Stream Ciphers encrypt individual characters (usually binary digits) of a plaintext message one at a time, whereas *block ciphers* encrypt groups of several characters in one step. Implemented in hardware, stream ciphers tend to be faster than block ciphers. Which kind of cipher is more appropriate depends on the kind of application. Stream ciphers are preferred, if buffering is limited or received characters must be processed immediately. Another advantage of stream ciphers is that they have no or limited error propagation which is useful when transmission errors are highly probable [9].

A stream cipher generates what is called a keystream (a sequence of bits used as a key). Encryption is accomplished by combining the keystream with the plaintext, usually with the bitwise XOR operation. The generation of the keystream can be independent of the plaintext and ciphertext. The result is what is called a *synchronous* stream cipher. If the key stream depends in some way on the data and its encryption the stream cipher is called *self-synchronizing*. Most stream cipher designs are for *synchronous* stream ciphers.

Synchronous stream cipher model

In figure 5.1, we see the model of a stream cipher. The plain text *P* is transformed to a cipher text *C* through applying functions that depend on several keys. The transformation is a bit wise *xor*-function. The block that defines the key-bit is a finite state machine. Key 1 and key 2 change the behaviour of the state machine. Its output is filtered with the influence of key 3. The keystream's state machine on the receiver's side works in a synchronous way. Errors of a bit in a stream,

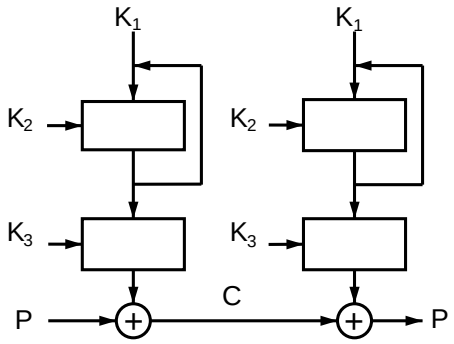

Figure 5.1: Stream cipher model

regardless whether it is the key stream or the cipher text stream, do not influence the following bits since every bit is computed individually. Two examples for stream ciphers are *RC4* which originally was a trademark of *RSA Security* and software-optimized encryption algorithm (SEAL). SEAL was introduced in 1993. A specific goal of SEAL was the efficient implementation in software, in particular, for 32-bit processors. SEAL is a *binary additive* stream cipher which means that keystream, plaintext and ciphertext are binary digits that are *xored* with one another.

5.3.2 Block cipher

Symmetric key block ciphers are the basic building block for a lot of cryptographic mechanisms. They allow the construction of for example hash functions and stream ciphers.

As part of the cryptographic primitives block cipher can serve as a central component in data integrity methods, message authentication techniques, entity authentication protocols and confidentiality mechanisms.

A block cipher is a function that maps plain text blocks with a certain block length to cipher text blocks with the same length. The encryption key is a parameter of this function, the encryption function. To decrypt the cipher text, the encryption function is applied in the reverse direction. The decryption function therefore is the invertible mapping of the encryption function. Furthermore the encryption function must be one-to-one: each plain text block with a fixed key has to correspond to exactly one cipher text block. The function is therefore called *bijective*.

To encrypt messages that exceed this n-bit length there are several so called *modes of operation*. These specify the way the output and keys of one stage of the algorithm influence the input of the next one. Common modes of operation are: electronic code book (ECB), CBC, cipher feedback (CFB) and output feedback (OFB). The simplest mode is ECB, where the plain text blocks are enciphered independently from one another. For a detailed description of the modes of operation please refer to chapter 7.2.2 in [9].

DES

The most popular block cipher is the data encryption standard (DES). It was developed in the mid 1970's as the first commercial-grade modern algorithm with openly and fully specified implementation details. It is defined by the American Standard FIPS 46-2 [39]. DES encrypts plain text blocks of 64-bit size with the use of a 64-bit key. However, the effective key size is 56, since 8 bits are used as parity bits. Further details of the algorithm are left over to [9]. Attention has to be paid

to the choice of the key, because there are some *weak* and some so called *semi-weak* keys, which reduce the strength of the cipher.

A variant of DES is *3DES*. It is a more secure variant of DES. DES has an effective key size of 56 bit. Since this defines an upper bound to the key space and cipher operations are generally designed to be computational efficient, exhaustive key search is a widespread method to break DES. 3DES is a sequence of DES encryption, DES decryption and again DES encryption. The same modes of operations apply to 3DES. The encryptions respectively decryptions should be done with three different keys. However, it does not severely affect security if the same key is used for both encryption operations [9]. Thus, there are only two different keys.

5.4 Asymmetric cryptography

The primary problem with symmetric ciphers is not their security but the key exchange. Once the sender and receiver have exchanged keys, they can be used to securely communicate. The question is: Was the establishment of the key itself secure? Another problem is the number of necessary keys. If there are n people who need to communicate with one another, there is a need for $n(n-1)/2$ keys. This is only acceptable for a small number of people.

Asymmetric cryptography or public-key cryptography was introduced to solve these problems. An encryption algorithm based on public-key techniques is also called public-key cipher. A public key cipher uses a pair of keys for securely transmitting messages. The two keys belong to the person receiving the message. One key is a public key and may be given to anybody. The other key is a private key and is kept secret by the owner. A sender encrypts a message using the public key. Only the private key can successfully decrypt the message. This approach solves the key exchange problem inherent with symmetric ciphers. There is no need for the sender and receiver to agree upon a key. What is required is that before the start of the confidential communication the sender gets a copy of the receiver's public key. Here, only n key pairs are needed for n people to communicate privately with each other.

5.4.1 RSA public-key encryption

RSA Cryptography Standard One [32] is a very popular public-key cryptography standard. It can be used for *confidentiality* as well as *digital signatures*. The name of the system is derived from the starting letters of its three inventors: Rivest, Shamir, Adelman. Its security is based on the problem of integer factorisation: it is hard to factorise the euler ϕ-function of a number n if one does not know some special facts about n. At first, entities in a RSA public-key sytem must generate their key pair:

1. Generate prime numbers p,q
2. Calculate $n = pq$
3. Calculate $\phi = (p-1)(q-1)$
4. Calculate $1 < e < \phi$, such that $\gcd(e, \phi) = 1$
5. Calculate $1 < d < \phi$, such that $ed \equiv 1(mod\phi)$
6. Public key: (e, n), private key: d

Every party generates two large random prime numbers *p* and *q*, both about the same size. Typical values for the length of these numbers are 1024 or 2048 bits [34]. In the following the modulus n is computed by multiplying p and q. The next step is to calculate the euler ϕ-function. Now follows the actual generation of the public key e and the private key d. A random integer *e* is selected, such that the greatest common divisor of e and ϕ is 1. This means that e is relative prime to ϕ. Then, the extended euclidian algorithm is used to compute the unique integer d. The numbers d and e must fulfill the requirement $ed \equiv 1(mod\phi)$. The party's public key is the tupel (n,e), the private key is d.

When all parties have their key pairs, they can start sending encrypted messages to each other. The encryption process using RSA public keys looks as described below. It assumes two parties A and B. B encrypts a message m for A, and A decrypts it.

Encryption process from B's point of view:

- Obtain A's public key (n, e).
- Represent m as an integer in the interval $[1, n-1]$.
- Compute $c = m^e \bmod n$
- Send message c to A.

The message m is represented as one integer number and is divided into blocks. The number of each block has to be smaller then the modulus n. For instance, if the length of modulo n is 1024 bit then the blocks have a length of 1023 bit [34]. Then the cipher text is computed for every block according to the formula: $c = m^e \bmod n$. With applying this formula the data is encrypted and can be send to the destination, in this case A.

Decryption process from A's point of view:

- Calculate $m = c^d \bmod n$.

The decryption process is similar to the encryption process. A separates the enciphered data blocks and applies the same formula to them. Only the exponent differs. B used the public key e as an exponent to encrypt the data, whereas A uses its private key d.

The complexity of the algorithm can be decreased by choosing "smart" values for the public key e, for instance choosing values that allow a fast completion of the module exponentiation.

The security of the RSA technique is based on two problems. The first one is the problem of integer factorisation. This is the factorisation of n into the two primes p and q. The second one is the discrete logarithm problem. This is the problem of retrieving d from the formula: $m = c^d \bmod n$. The cipher text c can be computed with the public key. The problem is finding d for: $c^d = m \bmod n$.

5.4.2 Comparison between symmetric and asymmetric cryptography

Symmetric-key and public-key encryption schemes have various advantages and disadvantages. Some of them apply to both. This section highlights a number of these.

There are four advantages that symmetric cryptography has over asymmetric cryptography. Firstly, symmetric-key ciphers can be designed to have high data throughput. Some hardware implementations achieve encrypt rates of hundreds of megabytes per second. Software implementations reach throughput rates in the megabytes per second range [9]. A second advantage

is that keys for symmetric-key ciphers are relatively short. The key of DES for instance is only 64 bit long [39]. A third advantage is that symmetric-key ciphers can be used to construct various other cryptographic mechanisms. They include pseudo random number generator (PRNG)s, hash functions and computational efficient digital signature schemes. The fourth advantage is that symmetric-key ciphers can be assembled to build other strong ciphers.

Two of the disadvantages of symmetric-key cryptography were already mentioned in the section about public-key cryptography 5.4. The first one is that the key in a two party communication must remain secret at both ends. The second one is that in a large network there are many key pairs to be managed. A further disadvantage is that in a two party communication the key should be updated frequently. This imposes a risk to the key management.

The properties of public-key cryptography are also its advantages. Each party in a communication using public-key techniques has only to keep its private key secret. Depending on the mode of usage this key pair may remain unchanged over long periods of time [9]. Every party has its own key pair that is used for all communications with other parties. Therefore the number of necessary keys in large networks is much smaller than in symmetric-key scenarios. Public-key schemes have also advantages in the use with digital signatures. The key for the verification of a signature is typically much smaller than the one for the symmetric-key technique.

However, public-key systems have also disadvantages. They comply with the advantages of symmetric key schemes. The throughput rates for the most popular public-key encryption methods are much smaller than the best known symmetric-key schemes. The key sizes for public-key systems are typically much larger than those required for symmetric-key systems. Digital signatures are also significantly larger than authentication tags for data origin authentication from symmetric-key schemes. Furthermore, no public key scheme has been proven to be secure (the same can be said about block ciphers).

Hybrid cipher

As the last section showed, both encryption schemes have a number of complementary advantages and disadvantages. Public-key schemes reduce the number of necessary keys in a system and are a very effective tool to distribute keys. Symmetric-key schemes have faster encryption and decryption operations. The idea of hybrid ciphers is to combine the advantages of both techniques. The key exchange is done via public key cryptography and the actual encryption and decryption of data is performed by symmetric mechanisms. Since the key can be established securely, it can be created at each new session. The key is therefore also called a *session key* [10].

5.5 Digital signatures

Digital Signatures are defined in ISO 7498-2 [25] as: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of that unit and protect against forgery e.g. by the recipient. Thus, digital signatures are a method to bind the identity of an entity to data. This is an important procedure for authentication (entity authentication as well as data origin authentication), data integrity and non-repudiation.

Signing means generating a tag, called *signature*, with a secret piece of information that is only known to one entity. The entity which signs the data. A signature must be verifiable without the knowledge of the secret key used to create it. The two algorithms that are necessary for generating and verifying a signature are called a *digital signature scheme*. Generally digital signature schemes

can be separated into two classes (both use public key cryptography): schemes with appendix and schemes with message recovery.

Digital signature schemes with appendix require the original message as input for the verification algorithm. The original message is used to compute a tag that is appended to the message. The tag is only needed to verify the authenticity of the message. The process of signing consists of two steps. In the first step a hash function is applied to the message. A hash function is used because the generation of a signature is relatively slow in many schemes. The hash function reduces the message to a binary string of a fixed length. A required property of the hash function is *collision resistance* (see section 5.2). This allows the assignment of one message to one hash value. In the second step the hash value of the message is signed with the secret key.

In the verifying process the verifying entity uses the public key of the signing entity, applies it to the signature and compares the result with the hash value of the message in question. Digital signature schemes with appendix are the most common ones, in practice. They are typically applied to messages with arbitrary length. Two examples are digital signature standard (DSS) [40] and *ElGamal*.

In the digital signature schemes with message recovery the original message can be recovered from the signature. Its knowledge is not required to verify the signature. In the signing process one entity applies a publicly known *redundancy function* to the message and encrypts the result with its private key. The encrypted part is then the signature. The choice of the redundancy function is crucial to the security of the signature scheme.

The verifying party has to obtain the signature and the public key of the signing party. It then decrypts the signature with the public key and verifies the result. These schemes are in most cases used for messages with fixed length. *RSA* and *Rabin* are examples for this scheme.

In order to use a digital signature scheme in practice one needs a digital signature process. A digital signature process consists of the digital signature scheme as well as a method for formatting data into messages that can be signed. Two such processes are *ISO/IEC 9796* [24] and *PKCS#1*[32].

Chapter 6

Security analysis

This chapter investigates the VoIP architecture of INFONOVA regarding security. The conceived conclusions and methods as well as the research itself are built upon the previous introductory chapters.

The beginning of the chapter describes the task. It explains the explored VoIP architecture. The following part analyses possible attacks on protocol H.323 and the involved component's interfaces. The final part introduces three mechanisms for a security framework, *H.235*, *TLS* and *SRP*.

6.1 Task description

The underlying architecture is shown in figure 6.1. It consists of one BES, two GKs and three EPs. Calls in this architecture can be routed over one or over two GKs. The protocol used is H.323. The call model is the GK-routed model.

The center of the investigation is the GK with its interfaces to the BES, to other GKs and to other EPs. The protocol for the communication between GKs is Q.931. The protocol between the BES and a GK is a proprietary one. An important criteria for VoIP systems and therefore for this analysis is performance. A GK is the core part of a telephony system and must be able to handle at least 20 to 30 call attempts per second.In this regard security shall produce as few overhead in terms of execution time as possible to the existing system.

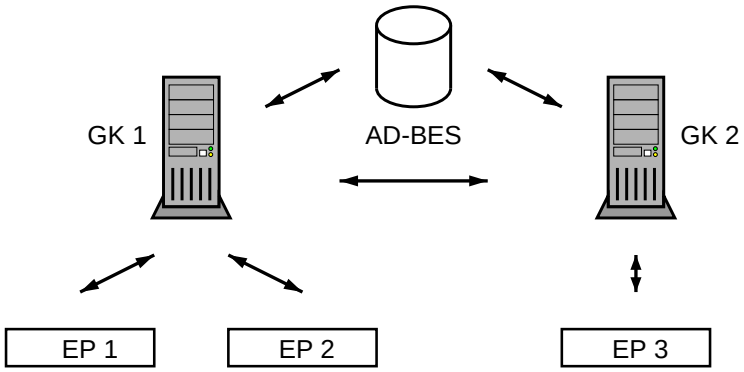

Figure 6.1: VoIP architecture

6.2 Threat analysis

6.2.1 Threat definition

Many threats to an IP telephony system are identical to those of any other system that is connected to the Internet. They include vulnerabilities of the network stack, of the operating system or of other services. These general vulnerabilities and attacks are covered in chapter 4. The threats analysed here concern the business model and the protocols between the components of a H.323 IP telephony system.

The business model is based on user subscription. Anyone who wants to use the telephony service has to be registered. The accounting is done according to the duration of the made calls. The main threat to the telephony system are people who try to call for free (also called phreaking). The following division was made to analyse possible threats:

- Manipulation of accounting data
- Direct call without the use of a GK
- Impersonation of an EP
- Impersonation of a GK towards a second GK
- Impersonation of the BES

An attack can be launched from both authorized and unauthorized users. Every subscribed person has one identifier that is used by its software (EP or terminal) to interact with the entities of the system.

6.2.2 Manipulation of accounting data

Every call made has to be accounted. The GK is the instance that administers the calls. It is responsible for gathering accounting data and for transmitting it to the BES. The BES collects and stores this data for later processing. The data structure for accounting data is called call detail record (CDR). For useful billing the CDRs must contain at least:

- Call duration: It consists of the start time and the end time of the call. The GK generates this value.
- CallID: It is a globally unique identifier assigned to each call. All call related data is indexed with it. Also this value is generated by the GK.
- UserID: It is a globally unique identifier for each authorized user. This value is defined at the time of the subscription.

CDRs are sent on the connection between a GK and the BES. There is one attack based on the data of the CDR. It is an active attack and aims to modify the value of the call duration value in the CDR. An attacker must have access to the data packets sent between the BES and the GK. He intercepts the data packet on their way from the GK to the BES, changes the value of the field containing the duration of the call and forwards the data packet to the BES. This exploit can be avoided by *peer entity authentication* in combination with *data integrity*.

6.2.3 Direct call

The accounting is based on the fact that all calls are routed over the GK. However, every VoIP terminal has the capability of directly calling another terminal without the use of a GK. The EP has to know the IP address of the destination, to be able to initiate a call with it. In this case, the GK does not recognize when a call is initiated. Therefore, these calls cannot be accounted. The identification of parties of a VoIP telephony system by some kind of identifier that do not correspond to the IP address does not help. Since the RTP traffic is always sent directly from one EP to the other one, only one regular call is needed to determine that IP address. EPs located on one subnetwork will therefore always be able to call each other directly. To prevent abuse on bigger networks, gateways that only allow call signaling traffic from GKs to pass have to be applied.

6.2.4 Endpoint impersonation

This section analyses the H.323 protocol for call initiation. It assumes that an attacker tries to impersonate an EP and shows the conditions for a successful attack. To initiate a call, an EP has to carry out a three stage process. The three stages are: sending an EP registration, a call admission and a Q.931 call setup message. The registration process is covered by the RAS messages *RRQ* and *RCF/RRJ*. Also the call admission belongs to the RAS protocol. The explicit messages for this procedure are: *ARQ*, *ACF*, and *ARJ*, respectively. The transport protocol of both the registration process and the call admission process is UDP. Thus, there is no real session between the EP and the GK for the RAS messages. An attacker could insert data packets into the connection at will. The actual call signaling is carried out by Q.931 which uses TCP. To launch an attack, an adversary can start its attack on different stages of the protocol. There are four possibilities for an exploit:

- Carry out the whole registration process
- Start by sending admission request
- Issue the Q.931 message setup
- Issue the Q.931 message setup with enabled *preGrantedARQ*

The result of the first case, carrying out the complete registration process, is that the GK accepts the attacker as the impersonated EP. The attacker can use every service that the subscribed user could. This is possible under two constraints. Firstly, the impersonated user is not registered at the moment of the attack. Secondly, if the used UserID is bound to a certain IP address, the attacker must reside on the network of the impersonated EP or on its network path to the GK. This is necessary since the attacker aims to establish a call and therefore needs to receive the responses of the GK.

If the attacker decides to impersonate an already registered user, he omits the registration request and starts by sending an admission request message. This is the second possible exploit. Here, the attacker has to use the identifier (UserID) of the user he wants to impersonate. He must also be able to receive responses from the GK. Thus, he must be on the same subnetwork than that user or on the network path from the GK to him.

The third possibility for an attack is to send the Q.931 message setup and skip precedent procedures. All Q.931 messages are identified by their CallID. The CallID used has to be a valid

one. CallIDs are generated during the call admission process. As a consequence the attacker has to take it from a call that was already permitted. However if the GK checks firstly the validity of the CallID and secondly whether it is already in use, the attack does not lead to a success.

An exception is the forth item of the list above, *pregranted ARQ*. Here, EPs do not request the permission of a call before trying to establish it by sending a setup message. In other words they skip the call admission procedure. Here, it is not possible for the GK to identify and assign setup requests to the corresponding user. Other methods, such as authentication, have to be used.

One example for such a method is to restrict the use of *pregranted ARQ* to certain groups of users. The access shall be as restricted as possible. Favorably it should only be used for IP networks with hard physical access. Another increase of security would be a binding of UserIDs to IP addresses. However, an attacker who resides on the same network as the user can eavesdrop its setup messages. In this case, the use of an authentication method is unavoidable to prevent attacks.

All the methods mentioned for attacks rely on the fact that there are no security mechanisms implemented. Nevertheless, the binding of important messages to properties like UserID, source and destination IP address makes an attacker's life much harder. Even more security can be reached by the use of passwords or shared secrets between EPs and GK. H.235v2-Annex D (see section 6.4) recommends the use of a HMAC function together with a password to secure[1] the connection.

If this authentication method is applied to all messages sent from an EP to the GK, the security depends on the strength of the password. The *dictionary attack* is a well known attack in this regard. The password should be as hard as possible to guess. In this case, the result of an attack only relies on the strength of the algorithm.

6.2.5 GK impersonation

As figure 6.1 shows, a call involves at least two EPs, one or more GKs and the BES. One EP (EP1) establishes a call to another EP (EP2). The call is routed over the first GK (GK1). If EP2 is administered by a different GK than the calling one, GK1 has to forward the call establishment request to the second GK (GK2). GK2 then contacts EP2. This call establishment request is a *Q.931* message, a *setup message*. GK1 finds out the "routing path" of the setup message by contacting the BES. The response from the BES contains information about EP1, EP2 and GK2. There are two scenarios where impersonating a GK can be successful:

- Impersonation against a second GK
- Impersonation against the BES

In the first scenario, the intent of an attacker is that a GK would accept a call setup request as if it was sent by a valid GK. There is no mechanism of authentication in Q.931 before the actual call establishment. This implies that a setup message sent to a GK that the GK could not identify as coming from a registered EP is coming from another GK. This is an unsatisfactory situation, since an attacker would only need to send a setup message to establish a call. A simple method to prevent this is to inform a GK about all other GKs in the telephony system. For instance, a GK could get all identifiers and IP addresses of other GKs at its startup when it registers with the BES.

[1] to secure in this context means entity authentication as described in section 5.1

However, here arises the problem that if the attacker is located between two valid GKs, he will be able to impersonate one GK towards the other one. Thus, the communication between two GKs requires mutual authentication. Moreover, if the two GKs are located on different networks there should be a kind of application level proxy server (see section 4.6.2) that only allows calls from authorized GKs to access the other network.

For the purpose of authentication, H.235v2, especially Annex D, has to be mentioned. H.235v2 also supports data integrity, which is also necessary for the connection. It is not enough to be sure about the identity of the sender of a message, if it is possible to change individual fields of the message. The used HMAC algorithm in H.235v2 provides both peer entity authentication and data integrity. However, the proposals of H.235v2 are restricted to the protocol. There are no descriptions regarding the security architecture or key generation, distribution and update. For EPs this is no problem, since they can be provided with passwords at the time of their subscription. The situation of GKs is more difficult. They need some kind of mechanism to be equipped with their passwords. This thesis works out a possible solution for key distribution in section 6.3.2.

The second possible scenario to impersonate a GK is against the BES. This can be helpful to discover UserIDs, IP addresses of EPs, GKs and their passwords. Because of this confidential data the proprietary connection between the GK and the BES should also be secured. Entity authentication as well as privacy is necessary. Another possibility is two divide the architecture into subnetworks. If the GK has two network interfaces there will be no need for further software measures to secure the GK-BES connection. The BES resides on a different network than the EPs and the two networks are completely separated from each other. Only the GK can access both subnets. Otherwise security measures, such as e.g. TLS should be implemented.

The last three sections of this chapter, H.235v2, TLS and SRP, will introduce methods to provide entity authentication and confidentiality.

6.2.6 BES impersonation

The BES only communicates with GKs. The protocol used is a kind of client/server protocol and it is a proprietary protocol. The GK requests information and the BES responds. The attack described previously (impersonation of the GK towards the BES) is also possible in the other direction: the attacker impersonates the BES.

Assuming that there is no authentication from the BES to the GK an attacker with the ability to intercept messages from the GK to the BES is able to send any kind of data to the GK as long as he respects the protocol. The attacker can forge his identity through intercepting a message from the GK to the BES and modifying, for example, the field that contains the password of an EP in the BES's response. He could even invent a new EP identity and fill the fields in the response with the corresponding values. These attacks lead to the conclusion that GK and BES needs to authenticate each other mutually. As in the previuos section a separation of the network into one subnetwork for the BES and one for the EPs solves this problem.

6.3 The architecture's components

The last section demonstrated possible threats for the protocol. One could see to what extent security services are necessary and indispensable. This section now depicts the role of the individual components in these scenarios and introduces security mechanisms to prevent vulnerabilities and exploits.

6.3.1 Endpoints

The security services that the connection between EPs and GK must support are peer entity authentication and data integrity. In other words, the GK has to be sure of the identity of the user and of the accuracy of the message sent. There are two possibilities for the implementation. The first one is a security method applied to every message that shall be sent. The second one is a concept that defines which EP has to apply what method. Such a concept is called a security policy. The term security policy can have different meanings [49]. Here, it will be referred to as the totality of rules for EPs to secure their call establishment requests.

The security methods that an EP shall apply can be known from the beginning. For instance, the utilization of a service requires a standard such as H.235v2. Another possibility is that the security methods are negotiated between the EP and the GK. The remainder of this subsection introduces both an example of how a security policy could look like and a way to negotiate the security methods.

Security policy

VoIP EP applications from different vendors support different security methods and different networks require different degrees of security. A security policy takes this into account to establish an appropriate degree of security. The GK is responsible for the application of the security policy. Therefore it has to know which security method every EP must use. This information usually is a table of data. It is a part of the configuration of the GK. Every user has got its entry in that table. Each entry will contain the necessary items to check the validity of a call request. These items can be:

- The IP address of the EP or its network address
- The protocols the EP uses
- The security methods the EP supports
- Whether the EP is allowed to use *pregrantedARQ*

According to the section *firewall* (4.6) an important principle is to restrict the access to resources as much as possible. The resource in this context is the permission to call. One example for the impact of the restriction of resources are IP addresses. IP addresses can be assigned to every user. The result is that each user can only make calls from the computer with this IP address. However, this disables EP mobility. It depends on the systems security policy whether this is allowed or not.

Table 6.1 shows an example of an entry in the security policy table. This example depicts the properties necessary to provide security. All messages during the call signaling process must be sent from or to the given user and IP address. The security methods to apply are specified in the fields where each protocol gets a security method assigned. The IP address avoids the risk of anybody calling with this UserID from another location. The other security methods *HMAC-SHA1-96* and *PwdHash* are used to prevent an EP impersonation by a different user on the same host. In general, the same method is used for all the protocols in H.323. The example of table 6.1 illustrates some possible methods. HMAC-SHA1-96 denotes the hash algorithm given in *H.235v2-Annex D*.

User ID	IP address	Pregranted ARQ connect allowed	Pregranted ARQ answer allowed	Protocol	Security method
Johann	10.1.13.30	YES	YES	IP	IP-Authentication
				RAS	PwdHash
				Q.931	HMAC-SHA1-96
				H.245	NoSecurity
				RTP	3-DES-EDE

Table 6.1: A security policy entry

Another common method is *PwdHash*, a hash of a password together with some additional information such as the time and UserID in order to prevent forgery or a reuse of the hash value. This method only provides peer entity authentication. Whereas HMAC-SHA1-96 provides peer entity authentication and data integrity. Both (PwdHash and HMAC-SHA1-96) are hash algorithms that are applied to messages in combination with a secret key. The result of that operation is appended to the message. The secret key and the fact that the hash operation is calculated over the entire message provide the message's authenticity.

One entry in the ARQ message is the CallID. The EP creates it and sends it the first time in that message. From then on the CallID is used in every H.225 message to identify the call. The GK uses the CallID to store reference information of the EP. If the ARQ is missing in the protocol the GK cannot store reference information for the initiated call to check whether it has been already permitted or not. Therefore, the GK does not carry out call establishment requests from EPs that did not request permission before sending the setup message. An exception is *pregranted ARQ*. With *pregranted ARQ* it is especially important to support entity authentication for Q.931. The reason is the missing ARQ message in the call signalling protocol.

Security method negotiation

A possibility to agree upon a security method is negotiation. Since the security method shall be utilized for call signaling, the negotiation process has to take place during the GK discovery phase. The request message issued by the EP is the GRQ (section 3.3.3). In the GRQ message the EP can set its authentication capabilities. The defined values for authentication mechanisms in H.225.0 in the sequence as they appear in the recommendation [1] are:

- Diffie-Hellman exchange
- Symmetric encrypted password
- Hashed password
- Certified signature
- IP security (IPSec)

- Transport layer security (TLS)
- Nonstandard
- Authentication at the BES

These entries in the field authentication mechanisms of the GRQ and GCF are only the names of security methods. The client shall include all the authentication methods it supports in the GRQ message. The GK selects one of them and includes it in the GCF message. There are no further explanations how to use these methods or implement them in the recommendation H.225.0. Therefore any implementation is to some extent proprietary (nonstandard) and has to be integrated for an EP and a GK. Moreover, not all of these methods support the required security services: a Diffie-Hellman key exchange alone does not support peer entity authentication. A simple hashed or encrypted password does not support data integrity. IP security and TLS can support both but need a lot of effort for the implementation.

The response of the GK is either a reply with a GCF, in case it accepts the security methods offered, or a GRJ. Because the GK may verify the message after deciding to accept the authentication method it proposes, the first authenticated message may already be the GRQ message itself. However, the response of the GK depends on the security policy. If the BES dictates certain authentication methods, the GK may insist on them. Here, the GK accepts the request of the EP only if the method proposed corresponds to the ones prescribed by the BES. A further possibility is that the GK supports a number of security methods that can be agreed upon, if a minimum level of security can be maintained.

6.3.2 GK

This subsection deals with the GK and the interface towards other GKs. It mainly covers *intra-GK communication* which is GK-communication within one AD. GK-communication between ADs is called *inter-GK communication*. The protocols in intra-GK communication are Q.931 and H.245, whereby H.245 can be tunneled over Q.931. Thus, the protocol of interest is Q.931. The topics this subsection will deal with are the authentication process, the key management and the role of the BES.

The intra-GK authentication process

The security services necessary in this section are peer entity authentication and data integrity. Another important matter are requests sent to the BES. Remote requests to the BES consume a lot of time and therefore shall be minimized. The setup message is the most important message in the protocol. It is the first message of Q.931 and there are no exchange mechanisms for security parameters before sending it. Thus, the setup message must contain information that ensures the identity of the sender and the integrity of the message. The recommendation for security for multimedia terminals [2] suggests among others HMAC-SHA1-96 as an algorithm to provide the security services necessary. The utilization of HMAC-SHA1-96 takes place on the application layer of the network stack. Other possibilities that fulfill the required security services are for instance IP security (IPsec) and TLS (6.5). Both are located on lower layers of the network stack. IPSec sets up on the IP layer and TLS resides between the transport layer and the application layer. However, HMAC-SHA1-96 is much easier to implement. Especially because it is already used to secure the RAS connection between GKs and EPs.

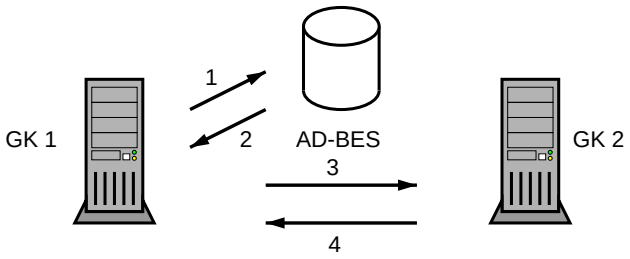

Figure 6.2: Authentication procedure: GK1 uses GK2's key

The keys of the GKs for the utilization with HMAC-SHA1-96 are, as well as the user keys, stored in the BES. The architecture of the system is the same that is already used throughout the entire section (see figure 6.1). One problem in GK-communication is the question which keys are used and how they are used. In the following three scenarios will be analysed:

- GK1 uses GK2's key for authentication
- GK1 uses its own key for authentication
- Mutual GK authentication

Each scenario has certain properties in regard to security. It is assumed that each GK has its own key. The differences between the individual scenarios are the way these keys are applied. Moreover, the first two scenarios assume only the authentication from one party to the second one. In the third scenario both GKs require an authenticated counterpart.

Another problem to be solved for GK-communication is the key generation. An EP gets the key necessary to use with HMAC-SHA1-96 when subscribing to the telephony service. The user has to enter it before he can use its terminal software. The situation is different for GKs. A GK has to know its own key from the beginning. Thus, it must be stored in a file. There are two possibilities to store the key: in plain text or encrypted. The disadvantage of a key stored in plain text is that it can be read if someone manages to achieve administrator rights on the computer. The disadvantage of an encrypted storage is that some person has to enter a key once to decrypt the actual GK-key.

GK1 authenticates with GK2's key

The first GK (in the following referred to as GK1) in a call signaling process that connects to a second GK (called GK2) must prove its identity. GK1 proves that it is authorized to connect to GK2 by possessing knowledge over GK2's key. It acquires this knowledge by sending a request to the BES (request 1 in figure 6.2). The BES responds with the key (number 2). GK2, in this case, does not have to connect to the BES since it knows its key. GK1 continues by sending the setup message (number 3) to the BES enlarged by an authenticator. The messages from GK2 sent to GK1 are subsumed in the figure under number 4.

This scenario implies that all other GK's must also connect to GK2 by using GK2's key. The knowledge GK2 has is that everybody knowing its key is authorized to connect to it. GK2 carries out the verification process with its own key. If the knowledge of GK2's key is enough for GK2 to accept setup messages from GKs, GK2 does not have to request data from the BES.

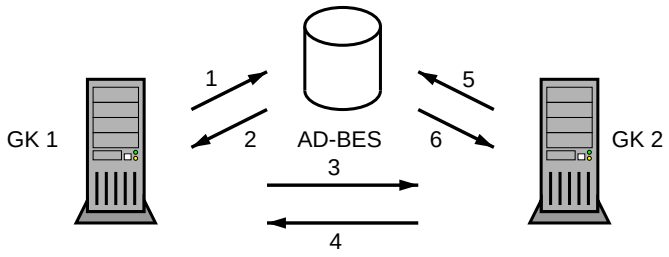

Figure 6.3: Authentication procedure: GK1 uses its own key

For further checks, GK2 needs additional information. Then GK2 must send a request message to the BES to get this data. One example for such information is the GK's IP address. In that case, the figure of the call signaling between the GKs corresponds to figure 6.3.

To increase performance a GK can store all the keys of the GKs it has connected to in its memory. This reduces the necessary connections to the BES. However, if the GK is attacked the risk of revealed keys increases.

GK1 uses its own key for authentication

Another possibility for our model is that GK1 uses its own key to authenticate to GK2. Before GK1 can send the call setup message to GK2, it has to send a request to the BES to get information about the destination ((1) in figure 6.3), for instance the IP address. After receiving the setup message from GK1 (3), GK2 has to aquire information about GK1. It will send a request message to the BES (5). The BES delivers the key and optionally some further data about GK1 (6). This data is used to verify the sender of the message, GK1. Since GK1 uses its own key this method provides peer entity authentication from GK1 to GK2. If GK1 uses HMAC-SHA1-96, it supports even data integrity. Also here the problem arises that the storage of resolved keys of other GKs in a GKs memory increases the risk of damage in case of an attack.

Mutual authentication

A visualization of the message flow of this scenario would deliver the same diagram as in figure 6.3. The difference is the content of the BES' replies and the application of the keys. Here every GK uses its own key to authenticate messages that it intends to send to another GK. The number of connections to the BES remains the same. GK1 has to send a request for necessary data, such as keying material of GK2 to the BES. GK2 also has to request data from the BES to verify GK1's message. Hence, mutual authentication can be reached with the same amount of resources.

The main problem of this method is the same than in the former one: in the end every GK will hold the keys of all other GKs. A better solution would be that every pair of GK has got their own, session dependent key for authentication, especially since all the keys are stored in the BES. The less keys the BES holds, the less keys can be adverted. The following protocol solves the problem that each GK holds all the GK keys after some time of operation and additionally reduces the amount of BES connections.

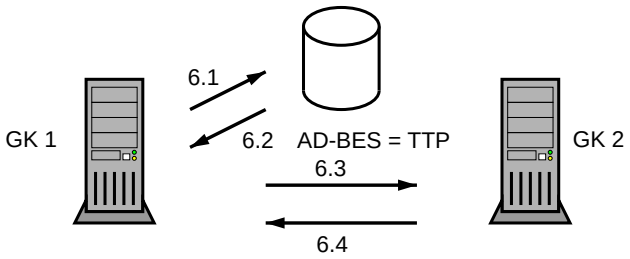

Figure 6.4: Kerberos-based protocol

Kerberos-based authentication protocol

The GK has to maintain a connection with the BES. This connection must provide peer entity authentication, integrity and secrecy for sensitive data. Kerberos, specified in RFC 1510 [28], provides peer entity authentication with an optional establishment of a shared key from one party to another one through a trusted third party (TTP). The TTP in our case is the BES. Figure 6.4 shows the protocol. GK 1 requests authentication data from the BES to authenticate at GK2. Already the first message sent to GK2 contains the necessary authentication information. The basic kerberos authentication protocol as outlined in [9] looks as follows:

$$GK_1 \rightarrow BES \quad : \quad GK_1, GK_2, N_{GK_1} \tag{6.1}$$
$$GK_1 \leftarrow BES \quad : \quad ticket_{GK_2}, E_{K_{GK_1T}}(k, N_{GK_1}, L, GK_2) \tag{6.2}$$
$$GK_1 \rightarrow GK_2 \quad : \quad ticket_{GK_2}, authenticator \tag{6.3}$$
$$GK_1 \leftarrow GK_2 \quad : \quad E_k(T_{GK_1}, GK2^*_{subkey}) \tag{6.4}$$

Notation:

- GK_1, GK_2, BES are the entities taking part in the protocol
- N_X is a nonce chosen by X. T_X is a timestamp from X's local clock.
- E is a symmetric encryption algorithm
- k is the session key(authentication key) chosen by BES, to be shared by GK_1 and GK_2.
- L indicates the lifetime of the session key.
- $ticket_{GK_2} \stackrel{\text{def}}{=} E_{k_{GK_2T}}(k, GK_1, L)$;
  $authenticator \stackrel{\text{def}}{=} E_k(GK_1, T_{GK_1}, GK^*_{1_{subkey}})$

GK1 starts the protocol by sending his identifier (ID), the destination's ID and a nonce to the BES (6.1). The BES responds with a *ticket* intended for GK2 and some encrypted data for GK1. This additional data is encrypted with the key shared between GK1 and the BES (6.2). Its data contains the session key k, the nonce N, the lifetime L of the session key and the destination ID of GK2. The nonce is provided to check the correctness of the message. The ID of GK2 actually indicates the connection for the valid session key. The ticket intended for GK2 contains the session

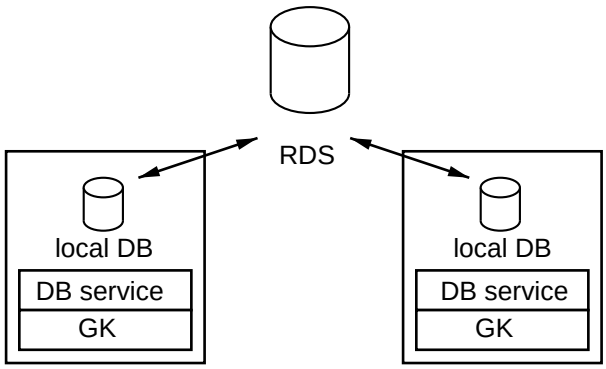

Figure 6.5: Architecture of two computers with a running GK and a running BES

key, its lifetime and an identifier of the party that request a connection to GK2. It is encrypted with the key that BES and GK2 share.

In the next step GK1 sends the ticket intended for GK2 as well as an authenticator appended to the setup message to GK2 (6.3). GK2 decrypts the authenticator using the session key and checks if the ID of the ticket matches the one from the authenticator. It then checks the validity of the authenticator by verifying the validity of the timestamp T_{GK_1} and tests its local time against the ticket's lifetime. When all tests are successful GK2 responds with the next message in the protocol, probably *call proceeding*.

If both messages 6.3 and 6.4 contain a subkey (e.g. Diffie-Hellman key half) a session key different from k can be established. Then even the BES cannot know the GKs' session key. This protocol can be applied to the security standard for H.323 connections: H.235. The ticket for GK2 as well as the authenticator can be included in Q.931 messages: the message field is called *cryptoTokens*. The authenticator can be generated according to H.235v2 - Annex D which would also ensure data integrity of the message.

6.3.3 BES

The BES realizes the service interface (SI) of the GK. All data needed to administer and run the system is stored there. This data includes EP- and GK-specific data such as passwords (shared secrets), IP addresses, aliases, security methods, configuration as well as call accounting information. It can be either requested all at once, for instance the security policy table for all EPs, or per request, when data is needed, for example an EP's key. The advantage of transmitting all configuration data at once is that connections to the BES are only necessary in case of an update of some data. The disadvantage is that one message contains a lot of sensitive information.

In general, the data the BES hosts is very sensitive. Therefore the implementation of the BES has to be done carefully in every area of security. In the following paragraphs some aspects of the BES's security are mentioned to stress its importance.

The SI can be either local or remote. Accordingly these two types are called local service interface (LocalSI) and remote service interface (RemoteSI). The LocalSI is a configuration file that holds all information. The structure of the RemoteSI is more complex. Figure 6.5 outlines its architecture. The central part of the architecture is the reference data server (RDS). It is the database where all the information is stored. Each host that runs a GK runs also a database (DB) service and a local copy of the database. In case of data updates in the central database, also the

local copies are updated. The DB service is the library that processes data requests from the GK to the database. It performs the corresponding queries to the database's tables. The connections between the GK and the DB service is at the moment a local socket. A local socket is a socket with the IP address $127.X.X.X$. This IP address is reserved for local connections. It cannot be accessed by other hosts on the network. Data sent to this address is not sent to the network. It is rather directly sent to the specified service on the local port.

DB service and the database could also be located on a different computer than the GK. Then, security considerations are important. A connection from the GK to the BES is in fact a connection between the GK and this service library, DB service. Because of simplification issues, the DB service and the database are callled BES and shown as one entity.

Physical Security

The sensitivity of the data stored on and exchanged with the BES has to be acknowledged through special concern regarding its security. It is important to look at security from a holistic point of view. The ISO standard ISO/IEC 17799 [26] and the "IT baseline protection manual" [13] are two documents that describe standard security mechanisms for all areas of an IT system. Examples for such security measures are the placement of the computer, the BES runs on, in a room that only grants access to authorized users or a secure operating system such as *OpenBSD*, where special concern during the development was put on security. Furthermore the machine hosting the BES shall run only the services that are needed for the operation of the BES, if possible only the database itself including a daemon for remote connections. The number of hosts that are allowed to connect to the server shall also be minimized and the protocol for remote administration shall be secure, preferably telnet over virtual private network (VPN) or a terminal on a serial interface.

Passwords

In the current implementation stage all authentication is done via passwords. The consequence is that all passwords reside in the database, which makes the BES a very vulnerable system.

In the secure remote password protocol (SRP), specified in RFC 2945 [52], a host does not have to store the plain passwords, it rather stores a so called *password verifier* along with a salt value and the username. This protocol abandons the need of plain text EP passwords in the database of the BES.

Because of the data stored in the BES and also because of all its data transmission contain sensitive information, the connection to the GK must be secured. One possibility that fulfills the requirements is public key cryptography. The advantage of it is that the BES would not have to store the passwords of all GKs, rather their public keys. This affects security in a positive way. The private keys would only be known by the individual GKs themselves. This measure in combination with SRP diminishes the sensitivity and thus the risk regarding data stored in the BES.

Network Security

The remote connection between the GK and the BES (RemoteSI) is at the moment of the work based on UDP. It is also restricted to local use. In order to secure a remote GK-BES connection the transport protocol must be changed to TCP. This is due to the fact that TCP offers a stateful reliable connection, in contrary to UDP. For instance, consecutive packets in TCP contain sequence

numbers to match and reassemble packets at the receiver's end. It is therefore much harder to manipulate the connection or insert packets.

Since the service of the BES requires the security services entity authentication, message integrity and secrecy, it is recommendable to use a standard protocol such as TLS for this purpose. TLS is a wide spread standard that can be used with several different authentication techniques. It negotiates some security parameters like passwords before the establishment of the actual connection. This negotiation, of course, requires some additional resources, especially time. The standard can be used because the call setup time restrictions do not apply during the registration phase of the GK at the BES. The reason is that only the establishment of the secure channel needs considerable amounts of time, especially when asymmetric cryptographic techniques are used. However a symmetric key for bulk data encryption can be negotiated at the connection setup. Then, only the process of bulk data encryption with the symmetric key enlarges the call setup time. This is due to the fact that the connection to the BES is already established.

6.3.4 Summary of the results

The following possibilities can be stated as results and as a starting point for further research: The interface between EPs and a GK requires peer entity authentication and data integrity. Password based authentication using *H.235v2-Annex D - Baseline security profile* provides this. Another possibility is a standard for securing connections: TLS. TLS using SRP also supports all necessary security services, peer entity authentication and data integrity. It even supplies privacy. The scenario with kerberos based authentication is one possible solution to secure the intra-GK connection. However, if TLS is implemented on the connection between the GK and the BES, e.g. using RSA key pairs, the next step, using the existing asymmetric keys for the intra-GK connection, lies close at hand. The impact to performance has to be considered and tested in this case.

Because of the sensitivity of the data exchanged on the GK-BES connection the application of a standard such as TLS is unavoidable. This method provides entity authentication as well as data integrity and secrecy.

The remainder of this section will describe the security methods mentioned until now. These are the recommendation of the ITU-T H.235v2 [2], as well as TLS and SRP.

6.4 H.235v2

H.235v2 is the ITU-T recommendation within the framework of the H.3XX series that incorporates security services such as *authentication* and *privacy*. The proposed scheme is applicable to both simple point-to-point and multipoint conferences for any terminals which utilize recommendation H.245 as a control protocol [2]. Secure real-time communications over insecure networks generally involve two major areas of concern - *authentication* and *privacy*. Recommendation H.235 describes the security infrastructure and specific privacy techniques to be employed by H.323 terminals to establish and maintain a secure communication. It further includes the ability to negotiate services and functionality in a generic manner and the ability to be selective concerning cryptographic techniques and capabilities utilized. The specific manner in which they are used relates to systems capabilities, application requirements and specific security policy constraints. H.235 supports different cryptographic algorithms, with varied options. Certain cryptographic algorithms may be allocated to specific security services.

Security Services	Call Functions							
	RAS	H.225.0		H.245	RTP		
Authentication	Password HMAC−SHA1−96	Password HMAC−SHA1−96		Password HMAC−SHA1−96			
Non− Repudiation							
Integrity	Password HMAC−SHA1−96	Password HMAC−SHA1−96		Password HMAC−SHA1−96			
Confidentiality							
Access Control					56−bit DES	56−bit RC2−com− patible	168−bit Triple− DES
Key Management	Subscription− based password assignment	Subscription based pass− word assign− ment	Authenti− cated Diffie− Hellman key− exchange	Integrated H.235 session key management (key distribution, key update using 56− bit DES/ 56−bit RC2−compatible/ 168−bit Triple− DES)			

Figure 6.6: H.235 Annex D: Baseline security profile

Authentication

The process of authentication verifies that the participating parties are in fact, who they claim to be. The most important component of the VoIP architecture is the GK. EP authentication is a major concern for it. There are two EP authentication mechanisms specified in recommendation H.235: One utilizes *a priori* known shared secrets and the other one uses public key based certificates. Authentication may be supported unidirectional or bi-directional. This authentication can occur on some or on all communication channels.

A further option for authentication is the use of other protocols specifically designed to convey security such as *IPSec* [27] or *TLS* [17]. These two protocols also shall be used to provide confidentiality on the communication channels.

For call authorization, the call establishment channel shall be secured. The call control channel (H.245 channel) conveys important information regarding securing the media stream. This information are messages to signal encryption algorithms and keys to establish secure media channels. In order to utilize H.245 in a secure manner the entire H.245 channel should be opened and negotiated in a secure way. The acquired information about the encryption keys and the used encryption algorithms can be used to securely establish logical channels (media channels). Different channels may thereby use different mechanisms.

H.235v2 includes two profiles to secure H.323 communications. The first one is called *baseline security profile* and covers basic security using password-based cryptographic techniques. The second profile deploys digital signatures. It is called: *signature profile*.

6.4.1 Baseline security profile

In the baseline security profile, EP and GK share a secret key which is used as basis for all cryptographic mechanisms. These keys are stored in the BES. On every EP registration, the GK requests the shared secret key with that respective EP from the BES. The GK uses this key to verify messages sent by the EP (also the registration message) and to compute tokens for messages that it sends to the EP. These tokens are values computed by an algorithm applied to the message together with the key. Here, this algorithm is a MAC. After the GK has calculated the token it appends it to the message and sends the message to the EP. The EP verifies the message that seems to come from its GK with its key and accepts it if the verification is successful.

The security provided by the baseline security profile works on a hop-by-hop basis. A hop is a trusted H.235 element along the communication path (e.g. a GK, multipoint control unit (MCU), proxy). Hop-by-hop means that at every hop the security information is verified and recomputed. On a path containing two EPs and one GK there are two such hops. One hop from the first EP to the GK and one hop from the GK to the second EP. After the first hop, the GK verifies the authentication information from the first EP, removes it from the message, adds authentication information for the second EP to the message and forwards the message to the second EP.

The functionality of the profile can be divided into two parts. The first is responsible for call signaling (includes H.225 and H.245) and is mandatory to be standard compliant. It offers authorization and integrity as security services. The second one, with the name *voice encryption profile* handles media traffic security. It is optional, because certain IP telephony environments already offer a certain degree of confidentiality (e.g. a dedicated telephony network operated on copper cables inside a building). However, the voice encryption profile may be applied if additional media confidentiality is desired. Figure 6.6 shows the security services supported for each protocol. The services for call signaling include authentication and integrity. Both are based on a keyed hash function, `HMAC-SHA1-96` (how an example token computed with HMAC-SHA1-96 looks like is illustrated in appendix B.3.2). The key is assigned at service subscription. Additional keying material exchange procedures may be applied during H.225. These procedures shall secure the key assignment and management for the media channel in H.245. The proposed cryptographic algorithms for securing the media channel are *56-bit DES, a 56-bit RC2 compatible cipher* and *a 168-bit Triple DES*. The baseline security profile mandates the fast connection procedure, as discussed in section 3.3.5. It does not prescribe confidentiality for call signaling. If desired, it may be implemented on a lower layer in the TCP/IP-stack. Confidentiality may be realized through other means such as *IPSec* (sets up on the network layer) or TLS. IPSec and TLS imply also authentication. The security features of H.235v2 concern the application layer. The baseline security profile is utilized in the call model of INFONOVA. For a detailed description please see the recommendation [2]. A disadvantage of this profile is the administration of all the shared secret keys. They have to be stored in a central place, which makes this one a critical part of the whole system.

6.4.2 Signature profile

The prerequisites for the utilization of this method are equal to those in the baseline security profile. The application of the GK-routed model and the fast connect procedure are mandatory. The digital signature model is an optional model in the standard. It introduces improved security through the use of digital signatures. Because there is no need of storing secret keys for all EPs this model is also more scalable and better suited for a global VoIP solutions [2].

Security Services	Call Functions			
	RAS	H.225.0	H.245	RTP
Authentication	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	
Non−Repudiation	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	
Integrity	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	SHA1/ MD5 digital signature	
Confidentiality				
Access Control				
Key Management	certificate allocation	certificate allocation		

Figure 6.7: H.235 Annex E: Signature Profile

Certificates in general give some assurance that the presenter of a certificate is who he claims to be. The intention is to authenticate the *user* not only the EP. With the usage of digital certificates a user can prove that he possesses the private key corresponding to the public key contained in the certificates. A verification of the certificates diminishes the risk of *man-in-the-middle attacks*.

The protocol describes the messages necessary for exchanging the certificates. But it does not specify the criteria by which they are mutually verified and accepted. Certificate policies are neither prescribed in this recommendation. However, applications using this framework may impose high-level policy requirements. But neither how these policies could look like nor how they are implemented is part of the standard.

Figure 6.7 shows a table with the security services of the signature profile. As in the baseline security profile, the security services authentication and integrity are supported. However, because a digital signature is assigned to a particular person, the actual person sending a message can be determined. It cannot claim that it did not perform that action. This property is called non-repudiation. The baseline security profile does not support it. The allocation of the certificates is part of the public key infrastructure which is not part of the standard.

6.5 TLS - Transport layer security

TLS, specified in *RFC 2246* [17], is a standard for building secure connections over the Internet. It is based on the services of a reliable transport protocol, such as TCP and offers its capabilities to the application. It can be seen as an intermediate layer between the transport layer and the application layer. TLS is a descendant of secure socket layer (SSL) version 3.0 where a client and a server exchange nonces to compute a session key. Version 3.0 was designed to correct a flaw where two parties could be adopted to a weak cryptosystem [42]. The latest version of the

protocol is TLS which closely resembles SSL 3.0, although they do not interoperate. TLS provides data integrity and privacy for the network connection of two applications. The protocol itself is composed of two layers: the *TLS record protocol* and the *TLS handshake protocol*. The lowest level of TLS, the TLS record protocol is layered on top of the transport protocol of the TCP/IP suite. An analysis of TLS can be looked up in [42].

Record layer protocol

The TLS record protocol provides connection security with two basic properties. The first one is a private connection. The protocol uses symmetric cryptography for this purpose. The second one is a reliable connection by an integrity check value that uses keyed hash functions.

The record protocol is a layered protocol. At each layer the messages may include fields for length, description and content. The protocol takes messages to be transmitted and fragments the data into manageable blocks. A block compression is optional. After the fragmentation (compression) the blocks are input to MAC hash functions. Finally the protocol sends the output of the MAC encrypted to the destination. On the reception of data blocks the protocol decrypts them, verifies them, (decompresses them) reassembles them and delivers them to the application. The entities performing these procedures are called *record protocol clients*. A record protocol client can for instance be the handshake protocol.

Handshake protocol

The TLS handshake protocol allows a server and a client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. It provides connection security with three properties:

- The peer's identity can be authenticated using asymmetric or public key cryptography
- The negotiation of a shared secret key
- A reliable negotiation

The handshake protocol is responsible for negotiating a session. It consists a.o. of the following items: *session identifier, the peer certificate, cipher spec, master secret*. The cipher spec consists of the bulk data encryption algorithm, the MAC algorithm as well as some cryptographic parameters. The master secret is a 48-byte shared secret between the client and the server.

As shown in figure 6.8 the client sends a `Client Hello` message to the server. The server must respond with a `Server Hello` message. These two messages are exchanged to establish security enhancement capabilities between the two parties. The exchanged attributes in these messages are: *protocol version, session ID, cipher suite, compression method* and two random values *clientHello.random* and *serverHello.random*.

The actual key exchange uses up to four messages: *the server certificate, the server key exchange, the client certificate and the client key exchange*. Currently defined key exchange methods exchange secrets which a range from 48 to 128 bytes [17].

After the `Server Hello` message the server will send its certificate, if the client requests authentication of the server. If a session key shall be established the server sends also a `Key Exchange Message`. If the server authenticated successfully, it may request a client certificate.

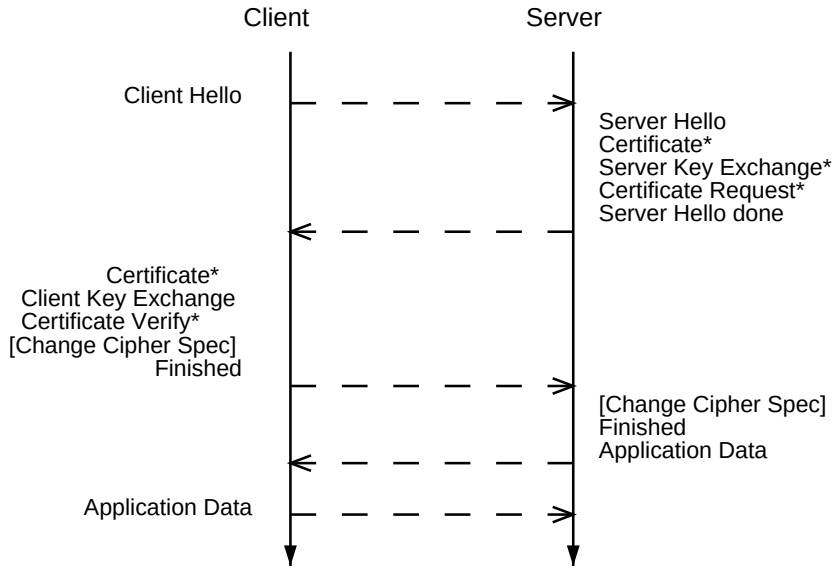

Figure 6.8: Message flow for a full handshake

The server indicates the end of this series of messages by sending the `Server Hello done` message. It will then wait for the client's response.

If the server requested the client's certificate the client will send the `Client Certificate` message, if a session key shall be established it also sends a `Client Key Exchange` message. At this point the client sends a `Change Cipher Spec` message. Immediately after this message the client sends `Finished`, already encrypted with the negotiated algorithms and parameters.

The server responds to the client's `Change Cipher Spec` message with a message from the same type. The message terminating the negotiation is the encrypted `Finished` message of the server. The handshake is then complete and the applications can start to exchange their data. There is also an abbreviated handshake protocol. Since this subsection shall only give a brief introduction to TLS it will not be described here. For further details see [17].

Connection termination

The end of a connection is signaled with the record type *alert*. Alerts can either be *closure alerts* or *error alerts*. An alert message with the label `fatal` results in the immediate termination of the connection. To terminate a connection either party can send a `close_notify` alert. The receiving party ignores all data it receives after this alert was sent. The sender closes its write side of the connection after sending this alert message. For a complete connection termination it is necessary that the other party responds with a `close_notify` alert.

6.6 SRP - Secure remote password protocol

SRP is a strong cryptographic network authentication protocol. It is based on the usage of passwords. Its security properties are the support of a secure session negotiation including authentica-

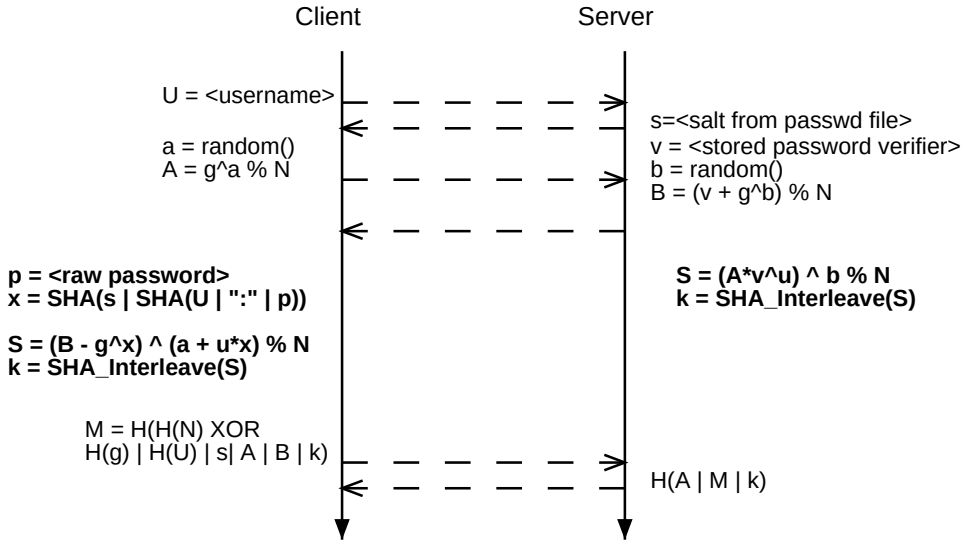

Figure 6.9: SRP-message flow

tion and privacy and that the servers stores a password verifier instead of the password itself.

This eliminates the weaknesses of conventional password based authentication mechanisms where servers store the passwords of clients that connect to its services. The key exchange takes place without the need of trusted key servers and a certificate infrastructure. The basis of the protocol are existing challenge-response mechanisms [52].

The password verifiers that the server must store have the following form:

$\{< username >, < passwordverifier >, < salt >\}$.

Password entries are generated as follows:

```
<salt> = random()
x = SHA(<salt> | SHA(<username> | ":" | <raw password>))
<password verifier> = v = g^x % N
```

The symbol | indicates string concatenation. Figure 6.9 shows the messages involved in the protocol and how the actual session key is computed. The first step of the client is to send its username to the server. The server responds with the salt value it stores in the password database. The client then generates A. It computes a random number, raises g to that power and modulos the field prime. Then it sends A back to the host. The host calculates B which is g raised by a random number concatenated with the public verifier and modulos it. After receiving A from the client the host sends B. The host never sends B before it actually received A.

If $B\%N$ is zero the client must stop the authentication process. The host does the same if $A\%N$ is zero. After the reception of A and B and their successful verification, both sides construct the shared session key based on the formula shown in figure 6.9 in bold letters. The parameter U is a 32-bit unsigned integer. It takes the value from the first 32 bits of the SHA1 hash of B.

Then the client and the server share a secure session key. To finish authentication, they prove to each other that their keys are identical. The server will calculate M using its own key k and compare it against the client's response. If the keys are not identical, the server aborts and signals

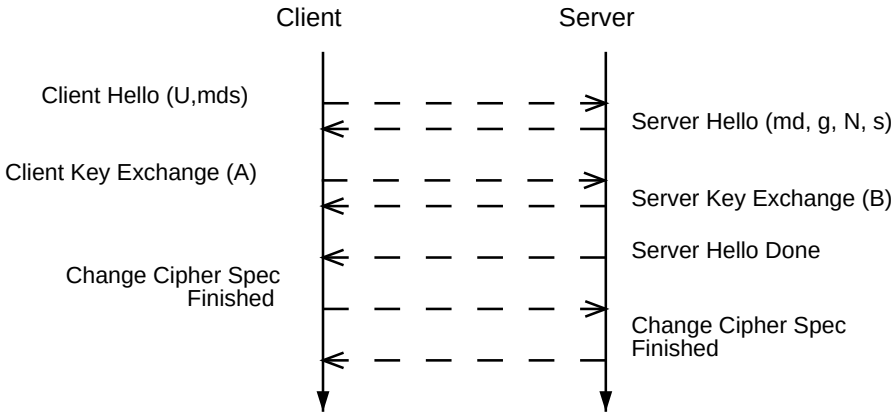

Figure 6.10: TLS-SRP-message flow

an error. This happens before the server answers the client's challenge. This mechanism saves the user's password from being compromised.

6.6.1 TLS with SRP

SRP can also be used together with TLS as a draft of the IETF shows [50]. This can be especially useful for the authentication of the EPs at the GK.

However to use SRP within TLS the message flow of TLS has to be adopted. Figure 6.10 shows it. The message flow starts with the `Hello` messages. The client extends the `Client Hello` message through appending the client's initial values. The server appends its initial values to its `Hello` message. The first modification to the protocol is that after the `Server Hello` message the client sends the message `Client Key Exchange` (this is mandatory in SRP). In the scenario before, the server issued the `Server Key Exchange` message.

After exchanging these messages the client sends the `Change Cipher Spec` message together with the message `Finished` to indicate the adoption of the pending connection states to the current ones. The server also sets the pending states to current, confirms the clients message by also sending a `Change Cipher Spec` message and a `Finished` message.

Chapter 7

Implementation

This chapter describes the practical part of the thesis. The task was to implement an authentication procedure for two GKs taking part in the same call. Each GK must be certain about the identity of the other GK. In other words: the task is to implement mutual GK authentication. This functionality had to be implemented in the existing GK software of INFONOVA.

7.1 Description of the task and prerequisites

The GK-routed call model is the basis for this case. As stated in section 3.3.5 all EPs are assigned to a H.323 zone and have to register at the GK that administers the zone. Moreover, the EPs share a secret key with the GK of their zone. A call from one EP to a second EP is either routed only over one GK, if both EPs are located in the same zone, or over two GKs, if the two EPs are located in different zones. The EPs do not recognize if there is more than one GK involved in the call establishment. The case that shall be considered for this implementation is that there are two EPs that are located in different zones. Thus, an established call has to be routed over two GKs.

The focus for this implementation lies in the interface between the two GKs. This is the protocol that shall be authenticated. The used protocol between the two GKs is *Q.931*. Q.931 is part of H.225 [1]. It is the call signaling protocol and is used to establish calls. In fact, not the two GKs that send each other Q.931 messages are establishing the call, the EPs do this. The GKs only forward the Q.931 messages exchanged by the EPs. Figure 7.1 shows the main messages that belong to that protocol. The first message in the call establishment is *setup*. It is the message that

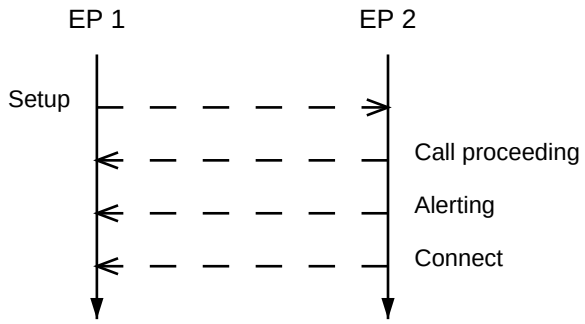

Figure 7.1: Call establishment using Q.931

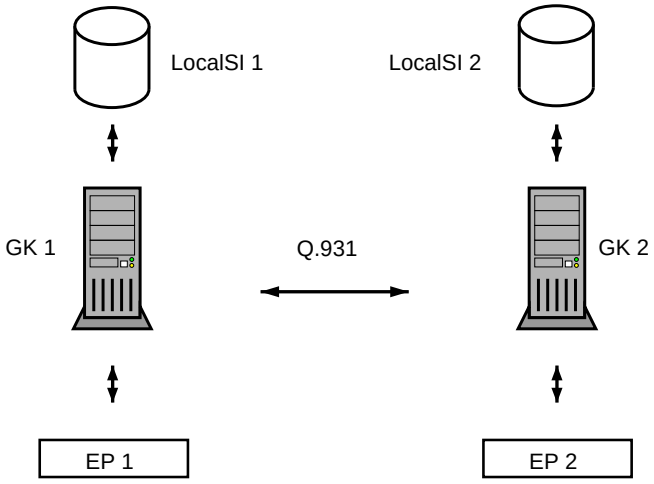

Figure 7.2: Call scenario for the implementation

initiates the call. Therefore it is important, that this message is only sent by authorized users. The messages after setup, until the call establishment is completed, are only responses from the called party. These messages are: *call proceeding*, *alerting* and *connect*. The ASN.1 definitions of these messages are given in appendix B.2.

The authentication for the call establishment is a hop-by-hop authentication. Each hop appends authentication information for the next hop. *End-to-end-authentication* of the EPs is not foreseen in this scenario. It is not of significant importance for the EPs that the IP telephony protocol incorporates end-to-end authentication. As in conventional telephony, identification of the other party is done by the users themselves. However, there could be cases, for instance in business conferences, where the two parties do not know each other and the identity of the other party is crucial. Anyhow, usually an EP's identity is only important to its GK (for the purpose of authentication).

Service interface

Every GK involved shall use its LocalSI to determine whether it allows an EP to register or to initiate calls. The LocalSI uses a file to store important configuration data. The implementation of authentication with the *RemoteSI* is beyond the scope of this work.

Important data stored in the configuration file are e.g. the passwords of the EPs that are "subscribed" to the service. The passwords shall be stored in an encrypted form. For this purpose a tool has to be implemented. Figure 7.2 shows the architecture for this scenario. Each GK has its own LocalSI withs its corresponding configuration file. The other components of the architecture are the same: two GKs and two EPs.

7.1.1 State of the GKs security framework

The security framework of the existing software includes authentication of the RAS protocol between EPs. The authentication method corresponds to recommendation H.235v2-Annex D, the baseline security profile. This authentication method is *HMAC-SHA1-96*, a HMAC algorithm [29] that uses SHA1 [38]. The result of the computation is appended as an authentication tag to the

message. There is a second authentication method which calculates a hash value of the shared secret key together with the UserID and the current time. This method is called *PwdHash*. ASN.1 examples for authentication tags of both methods are listed in appendix B.3.

The implementation of authentication procedures for the Q.931 protocol in GK communication (the task for the practical part) shall be extendable to EPs too. The assignment of the used security methods for EP authentication respectively GK authentication shall be listed in the configuration file. The entries in the configuration file specify the authentication method each entity has to use. They are a part of the system's *security policy* (see section 6.3.1).

7.2 Theoretical background

7.2.1 The authentication process

This section describes the authentication process by means of the establishment of a call. The mentioned process outlines the situation where the goals of the implementation are already met.

GKs that communicate try to initiate a call on the request of one of the EPs that are registered to it. The communication between the two GK is part of the *call signaling* to establish a call. This protocol is, as shown in figure 7.1, Q.931.

Call establishment including the authentication part of call signalling

A call from EP1 to EP2 is routed over GK1 and GK2. Hereby, EP1 is registered at GK1 and EP2 at GK2. The authentication of the registration procedure is done in the RAS messages. RAS also includes the EP's *admission requests* to launch calls.

After GK1 granted permission to establish a call to EP1, EP1 issues a *setup* message (the first Q.931 message) to GK1. GK1 retrieves information about the destination of the call from the BES (LocalSI). The data retrieved contains at least the information that GK1 has to route the call over a second GK (GK2) and the key that allows the calculation of an authentication tag for GK2. GK1 uses the shared secret key, computes the authentication tag, appends it to the message, and forwards the message to the destination EP's GK, GK2. GK2 receives the message from GK1 and verifies the authentication tag. If it can successfully check the authentication tag and identify the sender of the message as GK1 it forwards the message, after removing the authentication tag from GK1 and adding an authentication tag for EP2, to the destination EP, EP2. On a failure GK1 denies access to the destination EP and returns a *release complete* message.

An authenticated connection between two hosts is also called *leg* [2]. There are three different "legs" that belong to this call. The first one is the communication between EP1 and GK1, the second one is between GK1 and GK2 and the third one is the connection between GK2 and EP2. The protocols between an EP and a GK include the entire H.225.0 protocol (RAS, Q.931), the one between GK1 and GK2 only Q.931. Another thing to recognize is that there is no real end-to-end authentication. The two EPs do not authenticate each other. Rather the EPs authenticate at their individual GK and the GKs authenticate each other. Thus, the GK of the called party (GK2) trusts the GK of the calling party (GK1) that the calling EP (EP1) is authorized at GK1 to call EP2.

7.2.2 Knowledge of other GKs

A difficulty with authentication between the GKs is that a GK does not know which other GKs exist in the system. There is no protocol like RAS for GKs and call establishment in H.323 (Q.931) does neither provide information about other GKs. The situation for EPs is different. They register at a GK and request permission before initiating a call. There are two solutions to solve this problem. The first one is to let a GK know in advance which other GKs exist and which authentication method shall be used. This can be done e.g. through the BES at GK startup time. The second possibility is that the BES includes information about another GK every time the GK requests information about the call that shall be established. Both methods need a third entity that knows all GKs. In this case this is the BES. For the practical part of the thesis, the implementation of mutual GK-GK authentication, the second method was chosen. The reason is that the GK uses a LocalSI (a configuration file) to store configuration data. Thus, a "connection" to the BES does not impose a delay to the call establishment.

7.2.3 The authentication methods

An authentication tag is a tag computed with a shared secret key appended to a message. The shared secret key of an EP is determined during its subscription to the service and is known only to the EP and the GK administering the EP's zone. In the following, two methods to generate such an authentication tag are introduced. The first one is a hash of a password, the second one provides real entity authentication.

GK-PwdHash

This method simply hashes the password and some other values and appends the hash value to the message. The simple hash of the password is a weak authentication scheme. It does not provide timeliness or uniqueness. The tag can be captured by anybody who is able to wiretap the line. In the following, the attacker can take the tag and append it to any kind of message. But if the hash is not only computed over the password, rather over several fields of the message including source identifier, destination identifier and time, it is only possible to send a message with this tag in the tolerance period of the timestamp. This method provides better means of authentication. However, it does not provide data origin authentication, since the tag is not computed over the entire message. The ASN.1 message part is demonstrated in appendix B.3.1

HMAC-SHA1-96

This method provides peer entity authentication. Here, the authentication tag is computed over the complete message. Since the messages includes a source identifier and a destination identifier as well as a timestamp, all the necessary conditions for peer entity authentication are fulfilled. The method consists of a HMAC algorithm that uses SHA1 as hash function. The number 96 in the name of this method describes the number of bits of the output of the HMAC algorithm that are used for the hash value. Message details are illustrated in appendix B.3.2.

IP authentication

This is another authentication method that is used. Here, every EP is bound to one certain IP address. All messages from this EP have to be sent with this explicit source IP address. The

mapping of IP addresses to UserIDs is also contained in the configuration file (LocalSI) of the GK. Applying this method makes attacks that rely on user impersonation more difficult. The attacker has to be able to intercept packets sent to the forged IP address.

7.2.4 Password generator

This implementation uses the LocalSI to determine service-specific data such as user names, IP addresses and passwords. The LocalSI consists of a file that holds all configuration data. Since especially passwords are very sensitive information they shall not be stored in plain text. This idea leads to the development of a tool for the encryption of the passwords in the configuration file. The data in the configuration file has to be encrypted using a key. The encryption as well as the decryption process use symmetric cryptographic algorithms. *3DES* has been chosen as algorithm.

System start up

The person starting the GK has to enter a pass phrase. The key is derived from the pass phrase using PKCS#5 [31], a standard for password based cryptography. The GK uses the derived key to decrypt all the passwords of the other GKs. This process is part of the initialisation described in section 7.4.1.

7.3 The static model

This section discusses the static model of the implementation, the class structure. It describes the classes of the security framework and their relation to one another. Figure 7.3 shows all the classes that form the implementation. It includes the classes that I had to create to meet the requirements as well as the ones that are part of the original GK.

The classification of the classes is derived from their functionality: At first a GK needs `SecurityServices`. Among these, there must be an `AuthenticationService`. This `AuthenticationService` is different for EPs and for GKs. Therefore a separate `EPAuthenticationService` and a `GKAuthenticationService` are required. The task of an `AuthenticationService` is to add some means of authentication to a message that shall be sent. It shall also verify authentication mechanisms in messages received by the GK. These means of authentication are `SecurityMethods`. There can be several SecurityMethods: `NoAuthentication, IPAuthentication, HMAC-SHA1-96, EPPwdHash` and `GKPwdHash`. Three of these methods use hash functions. The hash function can either be *MD5* [37] or *SHA1* [38].

This small explanation reflects the class structure of the implementation. The classes included in the figure but not mentioned in the precedent paragraph do not directly belong to the security framework.

7.3.1 Use cases

Figure 7.4 shows all the use cases that can be specified for the implementation. They can be divided into three types: The first type is the initialisation of the security services. The second type is adding crypto tokens to a message. The third one is verifying crypto tokens. A crypto token is the ASN.1 specifier for the field that contains the additional information for computing

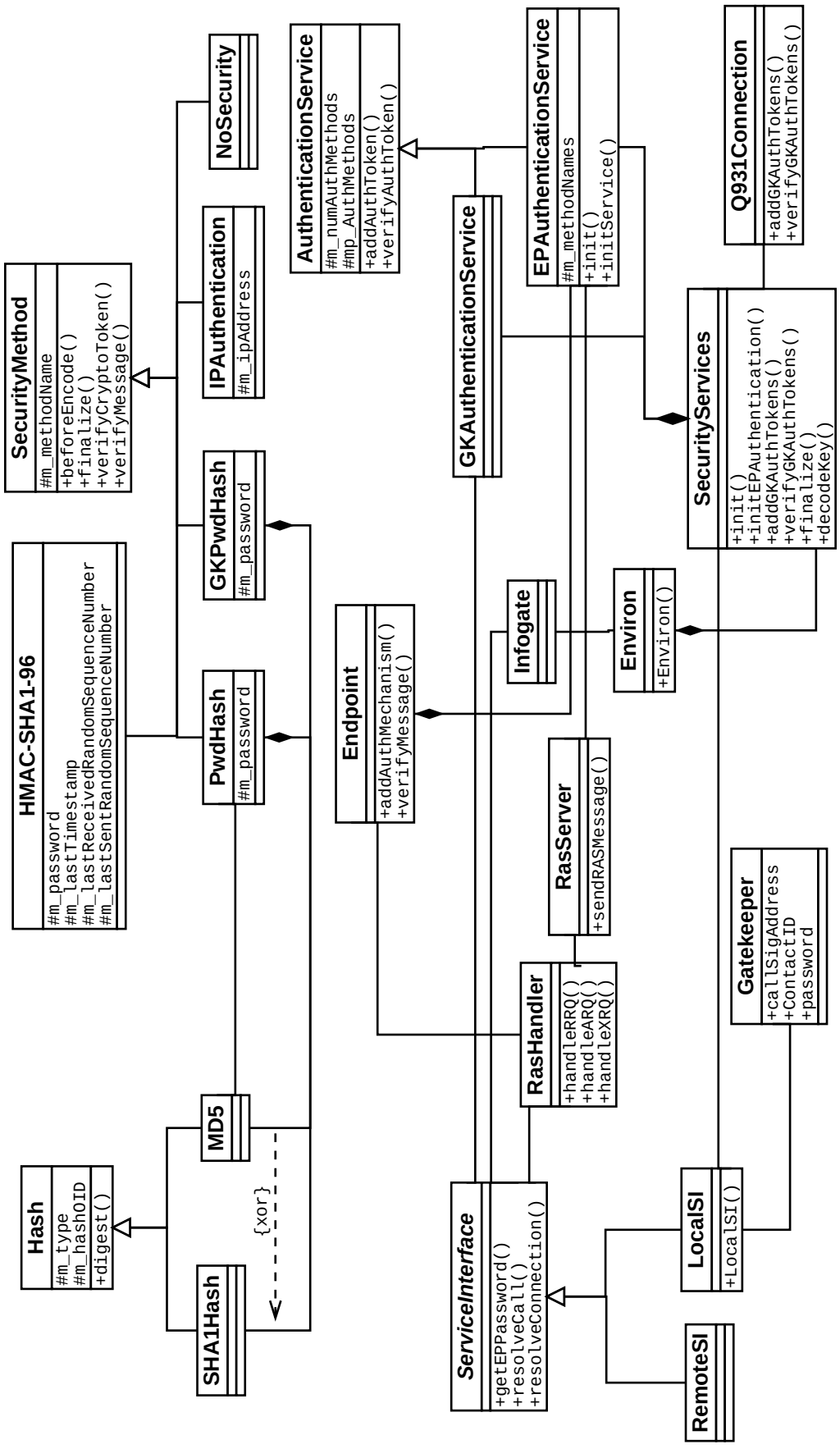

Figure 7.3: Class diagram

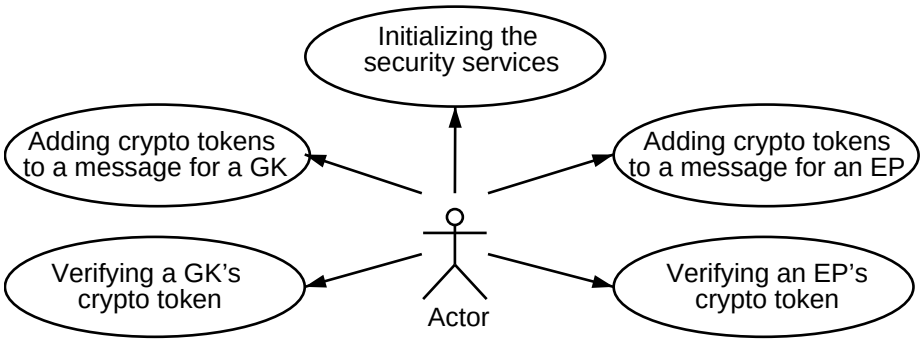

Figure 7.4: Use cases

the authentication tag as well as the authentication tag (hash value) itself. All three types of use cases are different for EP and GK.

The use case *initialising the security services* covers the initialisation of both the GK's authentication service and the EP's authentication service. It specifies what has to be done during the startup of the GK. The use cases *adding crypto tokens to a message* are responsible for computing the authentication tags for all authentication methods that shall be applied to the message. They append the authentication tags to the message that shall be forwarded to the other GK. The use cases *verifying a crypto token* verify all crypto tokens of a received message and specify the behaviour for retrieving the authentication tags from the message and verifying them. All use cases are listed in appendix C.

7.3.2 Necessary classes that do not belong to the security framework

There are some classes that interact with the security framework. These classes process received messages, send messages, or initialise the GK.

- Infogate: This class is the *root* class that starts all the threads and initialisation processes.
- Environ: It contains all the initialisation data from the configuration file; all data that does not concern the service interface.
- ServiceInterface: It is the basic class for LocalSI and RemoteSI and specifies the interface for the service invocation. The service can be data from the configuration file (LocalSI) or the BES (RemoteSI).
- LocalSI: This is the class that holds all the service-specific data if no BES is used. The data is retrieved from the configuration file.
- RemoteSI: It implements the remote interface to the BES. It was not used for the practical part.
- Gatekeeper: This class is a data structure holding some GK-specific data.
- RasHandler: This class processes all incoming RAS requests. Therefore it is the main class interacting with the security framework from an EP's point of view. In particular it initialises the EP's `EPAuthenticationService` and verifies all incoming messages. In order to send messages, `RasHandler` uses the service of `RasServer`.

- RasServer: It is responsible for sending RAS messages and accesses the security framework in order to add authentication information to the messages.

- Q931Connection: This class uses the security framework to verify incoming Q.931 messages from other GK's and to append authentication tags to messages intended to be forwarded or sent.

7.3.3 Classes of the security framework

SecurityServices

This class holds the `AuthenticationService` of the GK (`GKAuthenticationService`) and the initialisation of the class `EPAuthenticationService`. There is only one object of `GKAuthenticationService`. All the GKs that connect to that GK are verified by a method of this object. Therefore, all GKs have to use the same authentication methods. This empowers a homogeneous security level throughout the network. The GK creates an object of `EPAuthenticationService` for each EP that registers at the GK. After the initalization of that object by `AuthenticationService` the GK assigns it to the EP.

AuthenticationService

`AuthenticationService` is the basic class for `GKAuthenticationService` and `EPAuthenticationService`. It provides a container for an array of `SecurityMethod` and all the methods necessary to access them. Since the authentication service differs for EPs and GKs there are two further classes for each of them, `GKAuthenticationService` and `EPAuthenticationService`. Both are derived from `AuthenticationService`

GKAuthenticationService

This class implements `AuthenticationService` as it is needed for GK-GK communication. It is a part of `SecurityServices` and is initialised on the startup of the system. It is responsible for appending authentication tags to messages and for verifying them. A received message from another GK has to fulfill all the criteria that have been set during the initialisation. In other words all the initialised security methods (authentication methods) have to be present in the message and their verification must succeed. In order to send a message, all initialised security methods must add their authentication tag to the message. The security methods this class can contain are: `IPAuthentication`, `GKPwdHash` and `HMAC-SHA1-96`.

EPAuthenticationService

The `EPAuthenticationService` implements a container for the security methods used by EPs. Every EP gets an `EPAuthenticationService` assigned. This authentication service is initialised with the `EPAuthenticationService` in `SecurityServices`. All security methods initialised in `EPAuthenticationService` are integrated in the process of verifying a message or appending authentication tags to it. The security methods that this class can contain are: `NoSecurity`, `IPAuthentication`, `EPPwdHash` and `HMAC-SHA1-96`.

SecurityMethod

This is the basic class for all security methods. It provides the interface and a general implementation for verifying a message and appending crypto tokens to it. A `SecurityMethod` is meant to be one method to verify or add one specific authentication mechanism. Possible security methods are: `NoSecurity`, `IPAuthentication`, `EPPwdHash`, `GKPwdHash` and `HMAC-SHA1-96`.

NoSecurity

`NoSecurity` does not provide any security mechanism. This security method is only allowed for EPs.

IPAuthentication

This security method contains only an implementation for the case of a verification of a message. It checks the IP address of an EP.

PwdHash

This security method represents the *EP password hash*. The authentication tag of this method is the hash of the password together with some other data, such as a timestamp and UserID.

GKPwdHash

This class embodies the method introduced in 7.2.3. The hash value of the authentication tag can be computed either with *MD5* or with *SHA1*.

HMAC_SHA1_96

This class implements the authentication method of *H.235v3-Annex D: The Baseline Security Profile*.

Hash, MD5, SHA1Hash

The class `Hash` is the basic class for *MD5* and *SHA1Hash*. As the names indicate, MD5 implements the MD5 hash function and SHA1Hash SHA1.

7.4 The dynamic model

The static model illustrates the different classes of the architecture and describes them. What it cannot show is how these classes interact. This is covered in this section. It describes the interactions between actual objects of the classes mentioned in the static model. These interactions include the initialisation and authentication procedures.

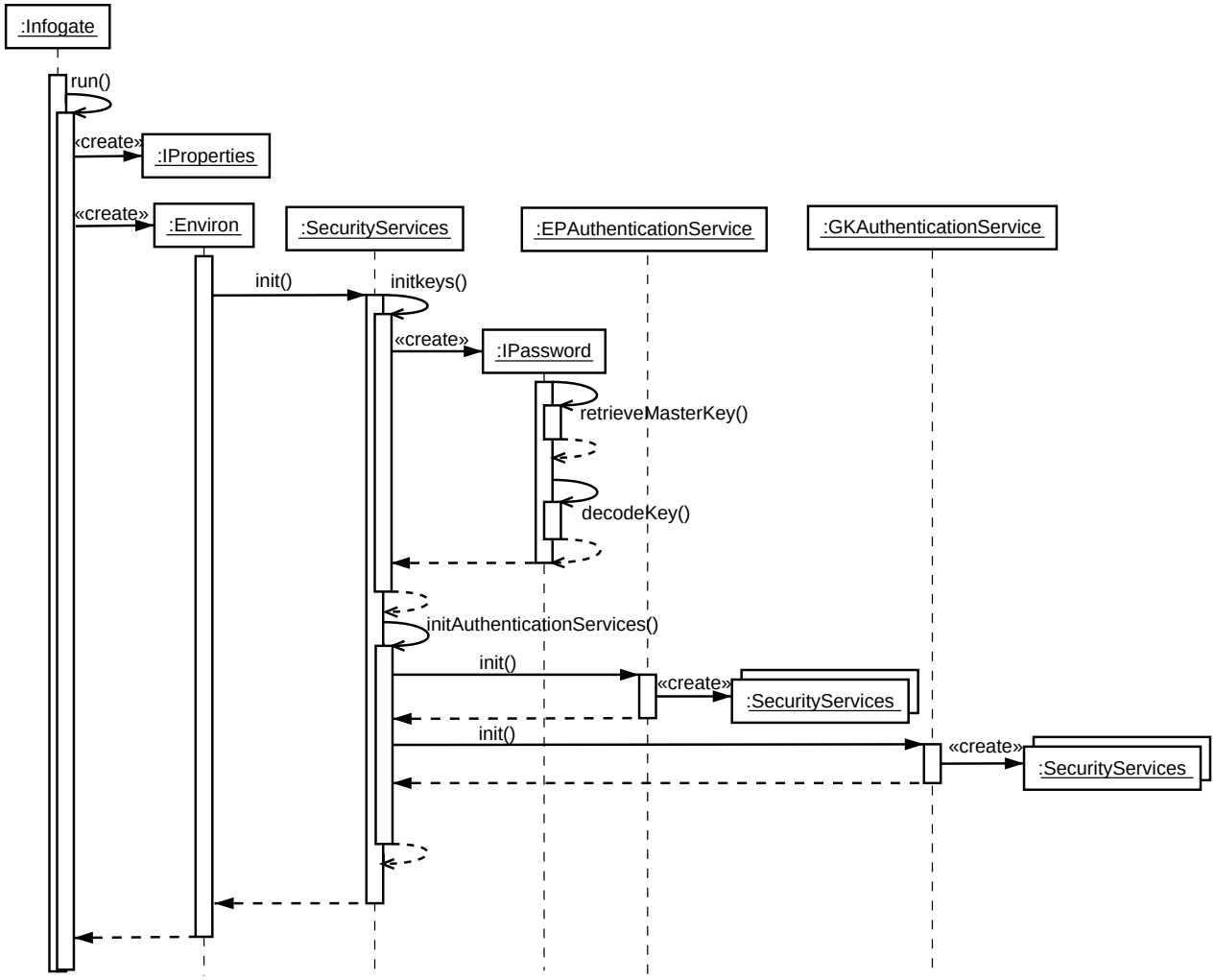

Figure 7.5: Initialisation of the security services

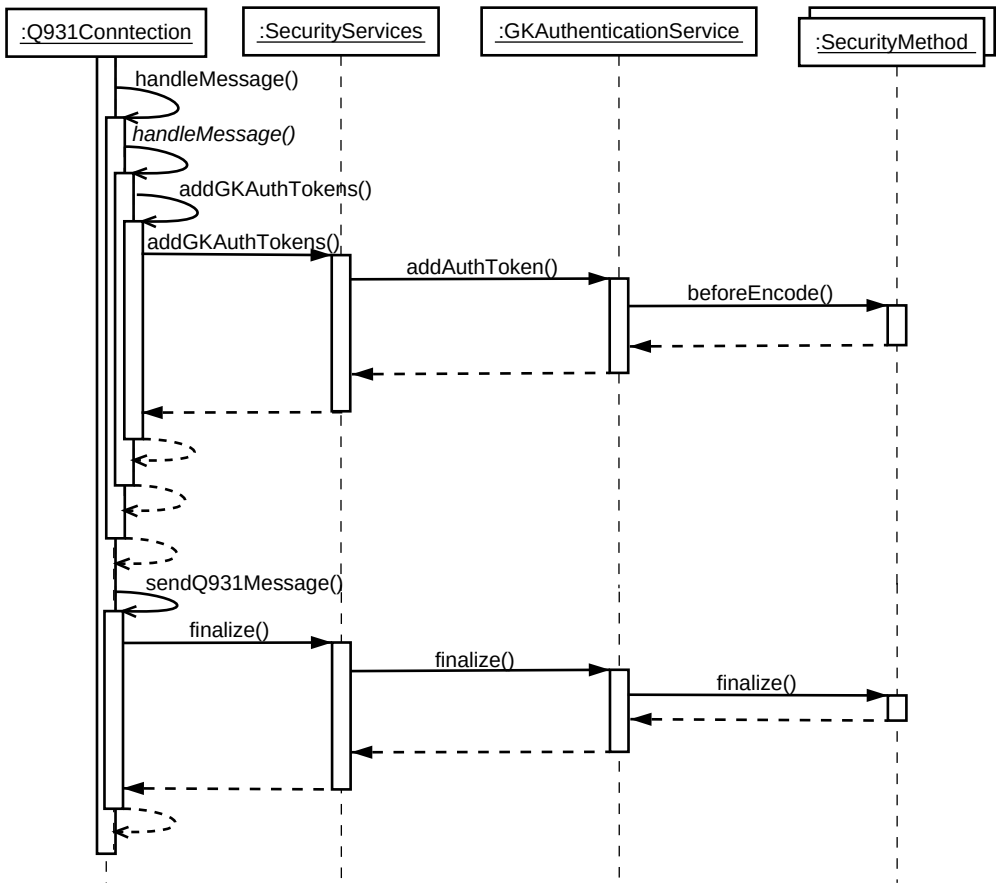

Figure 7.6: Adding a crypto token of a GK to a Q.931 message

7.4.1 Initialisation of the GK's authentication service

This process initialises the object `SecurityServices`. Figure 7.5 illustrates it. The initialisation of `SecurityServices` takes place at the startup of the GK. It is carried out by the main class: `Infogate`. The first step is to read the system's properties from the configuration file. As stated previously the configuration file is the LocalSI. The passwords for EPs and GKS are stored encryptedly.

The next step is to "build" the systems environment. This means to initialise the necessary parameters and services. One of these services is `SecurityServices`. In the method *retrieveMasterKey()* the person who started the system is prompted for the pass phrase of the so called *master key*. Then, the GK derives the master key from the pass phrase (which is also done in *retrieveMasterKey()*). The following step is the decryption of all the GKs' keys. The GK does this in the method *decodeKey()* which is called for each GK separately (in a loop). Finally, the GK can start to configure the authentication services. Therefore, it invokes the *init()* method of each authentication service. The required parameters are the ones read from LocalSI.

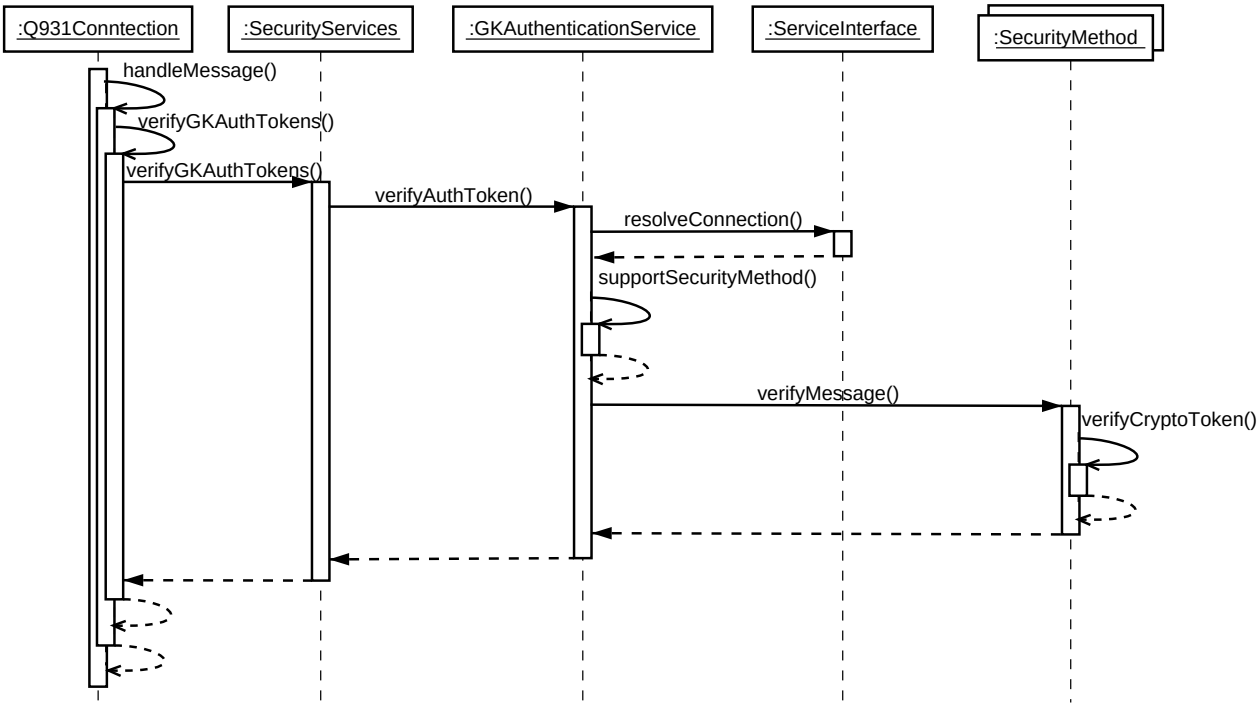

Figure 7.7: Verification of a GK's crypto token sent in a Q.931 message

7.4.2 Appending the GK's authentication tag

This case occurs whenever a GK shall forward a message to another GK. It concerns the Q.931 protocol and is shown in figure 7.6. The class handling the Q.931 connection of the GK is called `Q931Connection`. To calculate and append an authentication tag to a message it calls the method *addGKAuthTokens()* from the object `SecurityServices`. The process of authenticating a message is divided into two parts. The first step is to prepare the ASN.1 message and to add all the necessary information for the authentication tag. The part of the message that contains this data is called `cryptoToken`. This step is already sufficient for the security methods `PwdHash` and `GKPwdHash`, because they only add a hashed value to the message. `HMAC-SHA1-96` needs a second step. Since HMAC-SHA1-96 requires to hash the whole encoded ASN.1 message it is necessary to encode the message, compute the hash value and insert it into the encoded message. This message can then be sent to the destination. The method computing the tag and inserting the hash value is called *finalize()*. A difficulty is finding the location in the encoded message, where the hash value shall be added. For this purpose a special pattern, that cannot appear with normal encoding of message fields, is inserted into the message.

7.4.3 Verification of a GK's authentication tag

Whenever a GK receives a message from another GK it has to verify whether the source really is the one it pretends to be. Therefore the GK requests more information about the sender of the message from the BES. This procedure is shown in figure 7.7. The method *resolveConnection()* performs this request to the BES. The GK gets information about the source of the message. This information includes its IP address and its secret key. In the following the GK retrieves

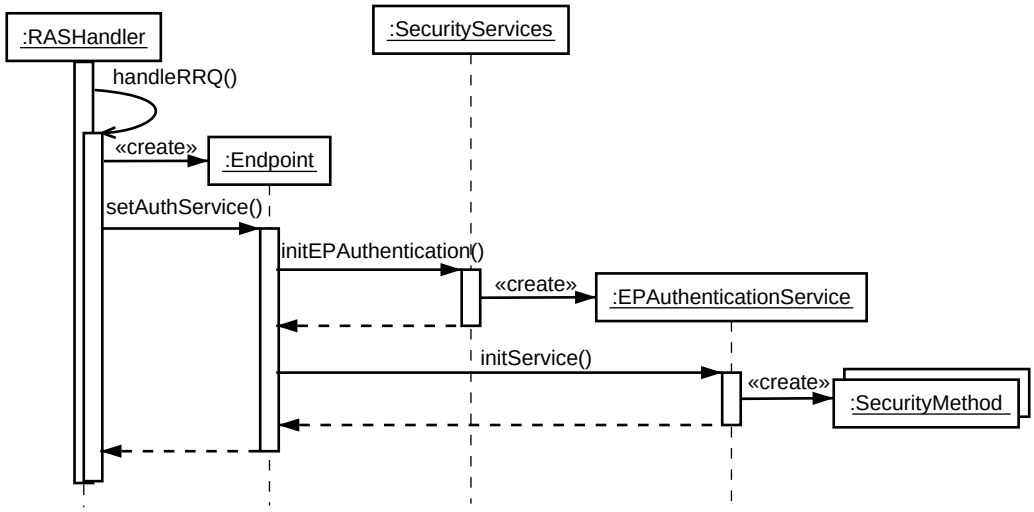

Figure 7.8: Initialisation of an EP's security services

the authentication tags from the message. It checks whether the (`cryptoTokens`) support the initialised security methods and verifies them. If the verification fails the GK returns a *release complete* message to the sender. In case of a success the GK removes the authentication tags of the source and forwards, after appending authentication tags for the destination EP, the message to that EP.

7.4.4 Initialisation of an EP's authentication service

Every EP must apply certain authentication methods. There are EPs with different authentication capabilities. The assignment of authentication methods to the EPs is done in the LocalSI. It assigns each EP its own "personalized" security methods.

On the startup, the GK initialises the authentication services. With the authentication services it initialises also the `EPAuthenticationService`. This process is depicted in figure 7.8. The `EPAuthenticationService` holds the security methods that the EP supports. The GK initialises the `EPAuthenticationService` right after creating the object. It invokes the EP's method *setAuthService()* for this purpose. In *setAuthService()* the GK invokes *initEPAuthentication()* to create an object of an authentication service. Afterwards it invokes *initService()* which initialises the security methods of the authentication service.

7.4.5 Verification of an EP's authentication tag

The verification of messages sent by EPs is similar to the process for the verification of messages sent by other GKs. There is one big difference: the EP's messages concern only RAS messages whereas the GKs messages affect Q.931 messages. In this case, the object dealing with the EP's RAS requests is `RasHandler`. `RasHandler` calls for every RAS request the corresponding method, *handleXRQ*. The X stands for the letter that abbreviates the type of RAS request. A list of the most common RAS requests and a link to further information can be found in section 3.3.3. In most methods the GK does not need to *resolve* the call (request further information about the EP from the BES)- except in one method *handleARQ()*. This is necessary because the EP wants

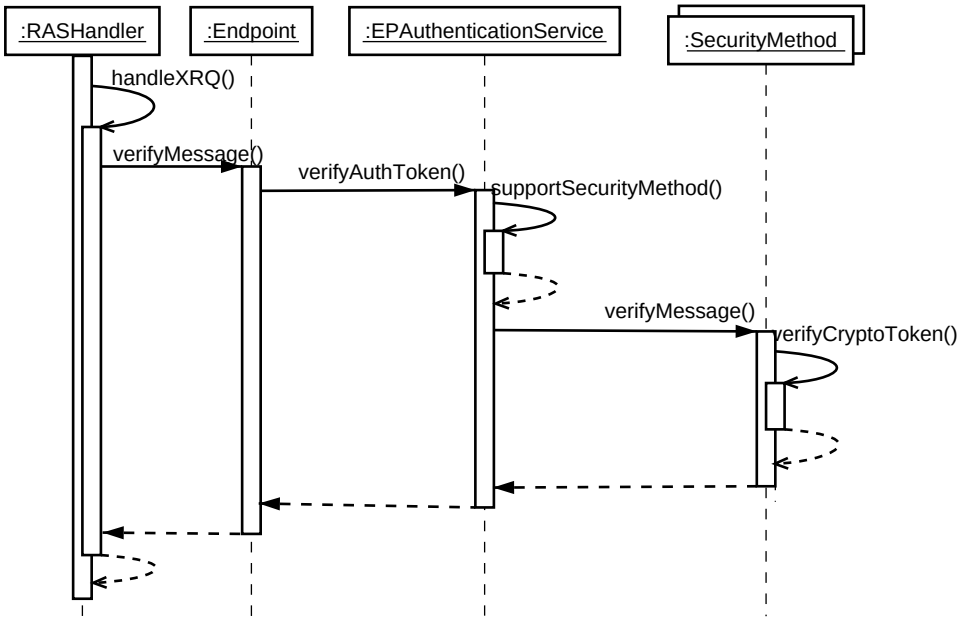

Figure 7.9: Verification of an EP's crypto token in a RAS message

to connect to another EP. This requires information about that EP. This data includes, whether the EP can be reached over this GK and the key of the entity that must be contacted. This entity can either be the destination EP itself, if it is registered at this GK or another GK. Figure 7.9 shows the general process of a verification. It does not take into account that *handleARQ* extends this functionality. In this case, the GK method *handleXRQ()* directly calls the EP's method *verifyMessage()*. This method invokes the method of the *EPAuthenticationService()* of the EP: *verifyAuthTokens()*. *VerifyAuthTokens()* checks for each `cryptoToken` of the message whether the security method represented by that token is supported. In case of success it verifies it. Otherwise it replies a *release complete* message to the sending EP. The reason (also called cause) of the reply message is *security denial*. A failure of the verification process also causes a *release complete* message.

7.4.6 Appending an EP's authentication tag

The GK carries out this case (illustrated by figure 7.10) if the previous use case, the verification of crypto tokens succeeded. The reason is that an authenticated RAS response message is only send upon a received and successfully verified RAS message. If the verification process fails, the returned RAS message (*release complete*) does not contain `cryptoTokens`. The object of a GK handling this case is `RasServer`. For every RAS message that shall be sent `RasServer` evokes the EP's method *addAuthMechanism()*. This method itself calls method *addAuthtoken()* from `EPAuthenticationService`. Method *AddAuthtoken()* appends the authentication tag of every supported security method through invoking method *beforeEncode()* for each security method. Here, it is also necessary to split the process into one part that modifies the ASN.1 message and one that modifies the already encoded message for computing the authentication tag. This last part (indicated by method *finalize()* in figure 7.10) is as in case 7.4.2 only needed by the security method *HMAC-SHA1-96*.

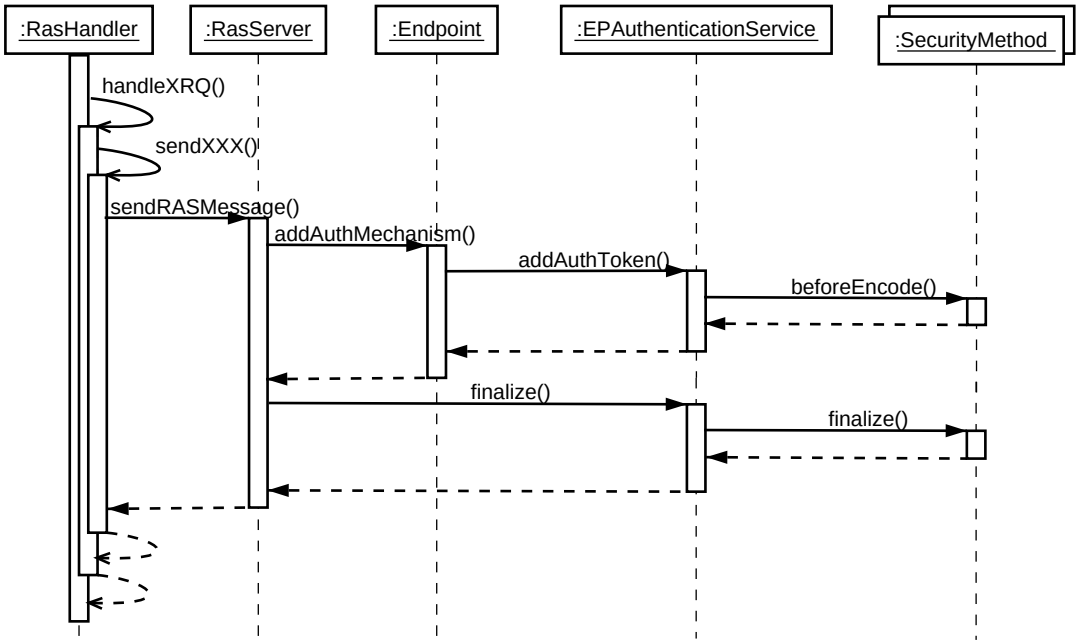

Figure 7.10: Adding of an EP's crypto token to a RAS message

Chapter 8

Conclusion

Security in VoIP Telephony Systems is a topic that involves many disciplines, for instance operating systems, Internetworking, VoIP protocols, cryptography and system security. For the development of a security architecture one must have a deep understanding of all of these areas. Moreover, there are other decisive factors that have a great impact on the final security architecture. Two of that factors that had an influence in this work are performance and the applied business model. The business model used was a GK-routed call model with subscription-based key assignment for EPs.

8.1 Goals

Since this thesis consists of a theoretical and a practical part there were two goals. The goal of the theoretical part was firstly to become acquainted with all the technical areas necessary to accomplish the task. Secondly, the used protocol had to be analysed according to existing threats, which was mainly the ability of attackers to call for free. Thirdly, research had to be done for possible security measures that fulfilled the needed requirements obtained from the analysis. The results, proposals for a security framework for the VoIP telephony system had to be developed.

The goal of the practical part was the implementation of mutual authentication on the connection between two GKs. The implementation should incorporate the results of the theoretical investigation regarding the interface between two GKs.

8.2 Challenges

One challenge was to develop an interdependent understanding of all the individual technical areas. The dependencies between TCP/IP, H.323, H.235, security, their properties, and the resulting actions are extensive areas of research on their own. A lot of experience is necessary to find the right mechanisms and to apply them in the right way. A holistic point of view is required for the analysis of the system. The proposed security measures shall not impose a lot of overhead in terms of execution time, since a GK shall be able to process about 20 call attempts per second. As many properties and fields of the protocol, such as the CallID or the source IP address, should be utilized to increase the security of the system.

8.3 Results

The investigation of the theoretical part yielded to the result that all interfaces require the security services authentication and data integrity. The interface between a GK and the BES additionally demands privacy because of the sensitivity of the transmitted data. The result for the proposals of the security framework must be seen for each interface or connection separately. The RAS protocol between EP-GK was already equipped with authentication and data integrity according to H.235v2-Annex D. If the use of *pregranted ARQ* is inhibited, this method already provides a certain level of security (see 6.2.4). The implementation of authentication of Q.931 according to H.235v2-Annex D (as it was done for the interface between GKs) was also integrated into the EP-GK interface. This allows now also the use of *pregranted ARQ*. Moreover, the proposed security policy from section 6.3.1 was implemented in the VoIP system during this thesis. It allows a flexible assignment of security methods that have to be applied by EPs. The security policy manifests itself as entries in the configuration file of the GK.

The authentication between two communicating GKs was part of the task of the thesis. There, the result was a working implementation for H.235 based authentication between two GKs. The key management has been kept simple since the GKs are using a LocalSI, which is a service interface realized by local sockets or UNIX sockets (up to now). A connection to these interfaces does not impose much overhead to the execution time. Therefore, it was not necessary to implement the Kerberos-based protocol for the GK-BES protocol, which was the proposal for securing the interface between the BES and the GK. This interface is still local and unless an attacker manages to get administration rights on the computer where GK and BES are located, there is no possibility to access that “traffic”. The protocol of the distributed database and its security are beyond the scope of this work.

8.4 For further studies

Further research has to be done mainly in the case of a remote BES (the BES is located on a different host than the GK). In that case, the connection is accessible via the network and therefore requires security mechanisms. One improvement would be to change the transport protocol for the GK-BES connection to TCP/IP because the insertion of data into a TCP connection is much more difficult than in a connection based on UDP.

The implementation of the Kerberos based proposal has to be worked out in greater detail. This proposal has advantages regarding the amount of connections to the BES and the key management between the BES and the GK. The number of connections is important if the BES is located remotely from the GK. The Kerberos-based protocol could be used to let two GKs negotiate their own session key, which would reduce the risk of a compromised BES. Further issues for research are procedures for a key update between the BES and the GK.

Another possibility is also the use of public key cryptography on that interface. In this case, standards as the DSS [40] or ISO 9796 [24] have to be researched regarding a possible application.

Appendix A

Abbreviations

AD administrative domain

AD-BES administrative domain - back end service

ACF admission confirm

ARJ admission reject

ARQ admission request

ASN.1 abstract syntax notation one

ATM asynchronous transfer mode

BER basic encoding rules

BES back-end service

CA certificate authority

CallID call identifier

CBC cipher block chaining

CCITT Comité Consultatif International de Télécommunication et Telegraphie

CDR call detail record

CER canonical encoding rules

CFB cipher feedback

CRHF collision resistant hash function

CSRC contributing source

DB database

DARPA Defense Advanced Research Projects Agency

DER distinguished encoding rules

DES data encryption standard

DoS denial of service

DDoS distributed denial of service

DMZ demilitarized zone

DNS domain name system

DSA digital signature algorithm

ECB electronic code book

EP endpoint

FDDI fiber distributed data interface

GCF gatekeeper confirm

GK gatekeeper

GRJ gatekeeper reject

GRQ gatekeeper request

HMAC hashed message authentication code

HTTP hyper text transfer protocol

IETF Internet Engineering Task Force

IP Internet protocol

IPsec IP security

ISDN integrated services digital network

ISO International Organization for Standardization

IT information technology

ITU-T International Telecommunication Union

LAN local area network

LocalSI local service interface

MAA message authenticator algorithm

MAC message authentication code

MC multipoint controller

MCU multipoint control unit

MDC modification detection code

MIME multipurpose Internet mail extension

MP multipoint processors

MTU maximum transmission unit

NAT network address translation

OFB output feedback

OSI open system interconnection

OWHF one way hash function

PBN packet-based network

PER packet encoding rules

PRNG pseudo random number generator

PSTN public switched telephone network

QoS quality of service

RAS registration admission status

RDS reference data server

RSA Rivest Shamir Adelman

RemoteSI remote service interface

SDP session description protocol

SEAL software-optimized encryption algorithm

SI service interface

SIP session initiation protocol

SSL secure socket layer

SSRC synchronization source

SRP secure remote password protocol

RTP real-time transport protocol

TCP transmission control protocol

TLS transport layer security

TTP trusted third party

UserID user identifier

UDP user datagram protocol

VoIP voice over IP

VPN virtual private network

Appendix B

ASN.1 messages

This chapter lists the most common H.225.0 messages with their most important fields. The three dots "..." indicate that the actual message contains fields that are not illustrated here.

B.1 RAS messages

B.1.1 GRQ

```
GatekeeperRequest ::= SEQUENCE
{
     requestSeqNum      RequestSeqNum,
     protocolIdentifier ProtocolIdentifier,
     ...,
     rasAddress         TransportAddress,
     endpointType       EndpointType,
     gatekeeperID       GatekeeperIdentifier OPTIONAL,
     endpointAlias      SEQUENCE OF AliasAddress OPTIONAL,
     ...,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     authenticationCapability
                        SEQUENCE OF AuthenticationMechanism
                        OPTIONAL,

     ...
}
```

B.1.2 GCF

```
GatekeeperConfirm ::= SEQUENCE
{
     requestSeqNum       RequestSeqNum,
     gatekeeperID        GatekeeperIdentifier OPTIONAL,
     rasAddress          TransportAddress,
     ...,
     alternateGatekeeper SEQUENCE OF AlternateGK OPTIONAl,
     authenticationMode  AuthenticationMechanism OPTIONAL,
     tokens              SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens        SEQUENCE OF CryptoH323Token OPTIONAL,
     ...
}
```

B.1.3 RRQ

```
RegistrationRequest ::= SEQUENCE
{
     requestSeqNum      RequestSeqNum,
     protocolIdentifier ProtocolIdentifier,
     nonStandardData    NonStandardParameter OPTIONAL,
     discoveryComplete  BOOLEAN,
     callSignalAddress  SEQUENCE OF TransportAddress,
     rasAddress         SEQUENCE OF TransportAddress,
     terminalType       EndpointType,
     endpointVendor     VendorIdentifier,
     ...,
     timeToLive         TimeToLive OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     ...
}
```

B.1.4 RCF

```
RegistrationConfirm ::= SEQUENCE
{
     requestSeqNum      RequestSeqNum,
     protocolIdentifier ProtocolIdentifier,
     nonStandardData    NonStandardParameter OPTIONAL,
     callSignalAddress  SEQUENCE OF TransportAddress,
     terminalAlias      SEQUENCE OF AliasAddress OPTIONAL,
     ...,
     endpointIdentifier EndpointIdentifier,
     ...,
     timeToLive         TimeToLive OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     ...,
     preGrantedARQ      SEQUENCE
       {
         makeCall                           BOOLEAN,
         useGKCallSignalAddressToMakeCall   BOOLEAN,
         answerCall                         BOOLEAN,
         useGKCallSignalAddressToAnswer     BOOLEAN,
         ...
       }
     ...
}
```

B.1.5 ARQ

```
AdmissionRequest ::= SEQUENCE
{
     requestSeqNum      RequestSeqNum,
     callType           CallType,
     ...,
     endpointIdentifier EndpointIdentifier,
     destinationInfo    SEQUENCE OF AliasAddress OPTIONAL,
     destinationCallSignalAddress
                        TransportAddress OPTIONAL,
     srcInfo            SEQUENCE OF AliasAddress,
     ...,
     callIdentifier     CallIdentifier,
     ...,
     gatekeeperID       Gatekeeper Identifier OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     ...
}
```

B.1.6 ACF

```
AdmissionConfirm ::= SEQUENCE
{
     requestSeqNum      RequestSeqNum,
     bandwidth          Bandwidth,
     callModel          CallModel,
     destinationCallSignalAddress
                        TransportAddress,
     ...,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     ...
}
```

B.2 Q.931 messages

B.2.1 Setup

```
Setup_UUIE ::= SEQUENCE
{
     protocolIdentifier  ProtocolIdentifier,
     h245Address         TransportAddress OPTIONAL,
     sourceAddress       SEQUENCE OF AliasAddress OPTIONAL,
     sourceInfo          EndpointType,
     destinationAddress  SEQUENCE OF AliasAddress OPTIONAL,
     destinationCallSignalAddress
                         TransportAddress OPTIONAL,
     ...,
     callIdentifier      CallIdentifier,
     h245SecurityCapability
                         H245Security OPTIONAL,
     tokens              SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens        SEQUENCE OF CryptoH323Token OPTIONAL,
     fastStart           SEQUENCE OF OCTET STRING OPTIONAL,
     ...
}
```

B.2.2 Call proceeding

```
CallProceeding_UUIE ::= SEQUENCE
{
     protocolIdentifier ProtocolIdentifier,
     destinationInfo    EndpointType,
     h245Address        TransportAddress OPTIONAL,
     ...,
     callIdentifier     CallIdentifier,
     h245SecurityMode   H245Security OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     fastStart          SEQUENCE OF OCTET STRING OPTIONAL,
     ...
}
```

B.2.3 Alerting

```
Alerting_UUIE ::= SEQUENCE
{
     protocolIdentifier ProtocolIdentifier,
     destinationInfo    EndpointType,
     h245Address        TransportAddress OPTIONAL,
     ...,
     callIdentifier     CallIdentifier,
     h245SecurityMode   H245Security OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     fastStart          SEQUENCE OF OCTET STRING OPTIONAL,
     ...
}
```

B.2.4 Connect

```
Connect_UUIE ::= SEQUENCE
{
     protocolIdentifier ProtocolIdentifier,
     destinationInfo    EndpointType,
     h245Address        TransportAddress OPTIONAL,
     destinationInfo    EndpointType,
     conferenceID       ConferenceIdentifier,
     ...,
     callIdentifier     CallIdentifier,
     h245SecurityMode   H245Security OPTIONAL,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     fastStart          SEQUENCE OF OCTET STRING OPTIONAL
     ...
}
```

B.2.5 Facility

```
Facility_UUIE ::= SEQUENCE
{
     protocolIdentifier ProtocolIdentifier,
     ...,
     conferenceID       ConferenceIdentifier OPTIONAL,
     reason             FacilityReason,
     ...,
     callIdentifier     CallIdentifier,
     ...,
     tokens             SEQUENCE OF ClearToken OPTIONAL,
     cryptoTokens       SEQUENCE OF CryptoH323Token OPTIONAL,
     ...,
     fastStart          SEQUENCE OF OCTET STRING OPTIONAL,
     ...
}
```

B.3 Examples of authentication methods

B.3.1 PwdHash

```
Message_UUIE ::= SEQUENCE
{
      .
      .
      .
      cryptoTokens = {
        elem[0] = {
          cryptoGKPwdHash = {
            gatekeeperID = h323_ID = {
              gate1.infonova.at
            }
            timeStamp = 2126623242
            token = {
              algorithmOID = 1 3 14 3 2 27
              params = <NULL>
              hash = {
                8F6A23DE19FEF7689AB19123A43B2323EF981234
              }
            }
          }
        }
      }
      .
      .
      .
}
```

B.3.2 HMAC-SHA1-96

```
Message_UUIE ::= SEQUENCE
{
      .
      .
      .
      cryptoTokens = {
        elem[0] = {
          nestedCryptoToken = {
            CryptoHashedToken = {
              tokenOID = 0 0 8 235 0 2 1
              hashedVals = {
                tokenOID = 0 0 8 235 0 2 5
                timeStamp = 1005902451
                random = 2126623305
                generalID = gate1.infonova.at
                sendersID = gate2.infonova.at
              }
              token = {
                algorithmOID = 0 0 235 0 2 6
                params = <NULL>
                hash = {
                  40956179AB971807F7CDA4318468ADCB341A4321
                }
              }
            }
          }
      }
      .
      .
      .
}
```

Appendix C

Use cases

C.1 Initialising the security services

- Goal - The GK shall have successfully initialised the security services
- Category - Primary. This case is carried out by each GK before it starts any threads. It is part of the initialisation. The initialisation process consists of reading and initialising configuration data (parameters) from the service interface.
- On success - The parameters of the security service are initialised successfully
- On failure - The startup of the GK shall exit with an error message
- Trigger- The case is triggered by the person invoking the GK. It is a part of the general initialisation procedure.
- Description of the process:

    1. the user is prompted for the “masterkey”
    2. the entered key is compared with a stored value derived from the key
    3. all the shared secret keys for the other GKs are decrypted with the masterkey. The initialisation of the GKs themselves is not part of this use case.
    4. the security services (authentication methods) have to be initialised for the EP and the GK. All the authentication methods that shall be used are retrieved from the configuration file.

- Alternatives for the actions above:

    – if the comparison of the stored value with the one derived from the entered password fails, the GK exits with an error message.
    – if any of the actions fail, no matter what reason, the GK exits with an error message.

C.2 Adding crypto tokens to authenticate at another GK

- Goal - To successfully add a crypto token to the message.
- Category - Primary. Every time an EP from another zone is called, the GK contacts a second GK to reach the destination.
- On success - The GK forwards the message with the appended authentication tag to the second GK.
- On failure - The GK issues an error message and returns a *release complete* message to the sender.
- Trigger - The use case is triggered by a call whose destination EP lies in another zone.
- Description of the process:
  1. The GK retrieves passwords necessary to calculate the authentication tag from the service interface.
  2. It calculates the authentication tag.
  3. It appends the tag to the message.
  4. The process with the numbers one to three is carried out for each initialised authentication method. Thus, each authentication method results in an authentication tag that is added to the message.
- Alternatives to the actions above:
  - In any failure case, the GK writes an error message to the logging facility and replies with a *release complete* message.

C.3 Verifying a GK's crypto token

- Goal - The successful verification of the crypto token of the sending GK.
- Category - Primary. Each time an EP from another zone calls an EP in the GK's zone, the connecting EP's GK routes the call to this GK.
- On success - The GK forwards the message to the destination EP (which is registered with this GK).
- On failure - The GK issues an error message and replies with a *release complete* message.
- Trigger - The reception of a message sent by another GK.
- Description of the process:
  1. The GK extracts all crypto tokens from the received message.
  2. It calls the service interface to get necessary data for the verification of the connecting GK (a.o. IP address, shared secret key).

3. The GK checks for every crypto token whether it supports the crypto token (was the corresponding authentication method initialised?) and verifies it.

- Alternatives to the actions above:
  - In any failure case the GK writes an error message to the logging facility and replies with a *release complete* message.

C.4 Verifying an EP's crypto token

- Goal - The successful verification of the crypto tokens of the sending EP
- Category - Primary. Every *request* message of an EP contains authentication tags that have to be verified.
- On success - Process the *request* message.
- On failure - The GK returns a *reject* message with the reason: *security denial*.
- Trigger - The reception of a request message.
- Description of the process:
  1. The GK extracts all crypto tokens from the received message.
  2. It calls the service interface to get the EP's authentication data (a.o. IP address, password).
  3. It checks for each crypto token, whether it supports the crypto token (was the corresponding authentication method initialised?) and verifies it.
- Alternatives to the actions above:
  - In any failure case the GK writes an error message to the logging facility and replies with a *reject* message with the reason: *security denial*.

C.5 Adding crypto tokens to a message intended for an EP

EPs have to authenticate their RAS messages. The process of adding crypto tokens to an EP's message is similar the one of adding crypto tokens to a GK's message. However, there are differences in the details of the implementation.

- Goal - To successfully add a crypto token to the message.
- Category - Primary. Each time an EP sends a RAS message to the GK, the GK has to add an authentication tag to the reply message intended for the EP.
- On success - The GK added the authentication tag successfully and returned the message to the EP. The GK has no means of controlling whether the EP accepted the message or discarded it.

- On failure - The GK issues an error message. This case should never appear. It is caused by an internal error of the GK. Here, no message is returned to the EP. The EP must retransmit its message and the GK tries to evaluate it again.
- Trigger - The use case is triggered by every request sent to the GK on behalf of a user's action.
- Description of the process:
  1. The GK builds the reply message and calculates the authentication tag.
  2. It appends the authentication tag to the message.
  3. It sends the reply message to the EP.
- Alternatives to the actions above:
  - In any failure case the GK writes an error message to the logging facility. There is no message returned to the EP.

Index

List of figures

List of tables

Bibliography

[1] Study Group 15. *Call Signaling Protocols and Media Stream Packetization for Packet-Based Multimedia Communication Systems*. Telecommunication Standardization Sector, ITU-T, November 2000.

[2] Study Group 15. *Security and Encryption for H-Series (H.323 and other H.245-Based) Multimedia Terminals*. Telecommunication Standardization Sector, ITU-T, November 2000.

[3] Study Group 16. *Proposed Revision of Recommendation H.322 - Visual Telephone Systems and Terminal Equipment for Local Area Networks which provide a guaranteed Quality of Service*. Telecommunication Standardization Sector, ITU-T, March 1996.

[4] Study Group 16. *Recommendation H.245 - Version 2 - Control Protocol for Multimedia Communication*. Telecommunication Standardization Sector, ITU-T, March 1997.

[5] Study Group 16. *Proposed Revision of Recommendation H.321 - Adaptation of H.320 Visual Telephone Terminals to B-ISDN Environments*. Telecommunication Standardization Sector, ITU-T, February 1998.

[6] Study Group 16. *Proposed Revision of Recommendation H.324 - Terminal for Low Bit-Rate Multimedia Communication*. Telecommunication Standardization Sector, ITU-T, February 1998.

[7] Study Group 16. *Proposed Revision of Recommendation H.320 - Narrow-Band Visual Telephone Systems and Terminal Equipment*. Telecommunication Standardization Sector, ITU-T, May 1999.

[8] Study Group 16. *Proposed Revision of Recommendation H.323 - Framework and Wire-Protocol for Multiplexed Call Signalling Transport*. Telecommunication Standardization Sector, ITU-T, September 1999.

[9] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press LLC, 1997. http://www.cacr.math.uwaterloo.ca/hac/

[10] Mike Ashley. *GNU Privacy Handbook*, 1999. http://www.gnupg.org/gph/en/manual/book1.html

[11] B. Preneel and P. van Oorschot. *MDx-MAC and Building Fast MACs from Hash Functions*, 1995.

[12] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message Authentication using Hash Functions - the HMAC construction. *RSA Laboratories' CryptoBytes*, Vol.2(No.1), Spring 1996. http://www-cse.ucsd.edu/users/mihir/papers/hmac.html

[13] Bundesamt für Sicherheit in der Informationstechnik. *IT - Grundschutzhandbuch*, Juli 2001.

[14] Daniel Collins. *Carrier Grade Voice over IP*. McGraw-Hill, 2001.

[15] Defense Advanced Research Projects Agency. *Internet Protocol*, September 1981. Request for Comment 791, http://www.ietf.org/rfc/rfc0791.txt

[16] Defense Advanced Research Projects Agency. *Transmission Control Protocol*, September 1981. Request for Comment 793, http://www.ietf.org/rfc/rfc0793.txt

[17] T. Dierks and C. Allen. *The TLS Protocol*, January 1999. Request for Comment 2246, http://www.ietf.org/rfc/rfc2246.txt

[18] M. Handley and V. Jacobson. *SDP: Session Description Protocol*, April 1998. Request for Comment 2327, http://www.ietf.org/rfc/rfc2327.txt

[19] John D. Howard. *An Analysis of Security Incidents on the Internet*. PhD thesis, Carnegie Mellon University, April 1997. http://www.cert.org/research/JHThesis/Start.html

[20] International Organization for Standardization. *Approved Algorithms for Message Authentication - Part 2: Message Authenticator Algorithm*, 1992. ISO/IEC 8731.

[21] International Organization for Standardization. *OSI - Reference Model: Part 1: The Basic Model*, second edition, November 1994. ISO/IEC 7498-1.

[22] International Organization for Standardization. *IT - OSI - Specification of Abstract Syntax Notation One*, first edition, Mai 1999. ISO/IEC 8824.

[23] International Organization for Standardization. *IT - OSI - Specification of Basic Encoding Rules for Abstract Syntax Notation One*, first edition, Mai 1999. ISO/IEC 8825.

[24] International Organization for Standardization. *IT - Security techniques - Digital Signature Schemes giving Message Recovery - Part 2*, first edition, December 2000. ISO/IEC 9796-2.

[25] International Organization for Standardization. *OSI - Reference Model: Part 2: Security Architecture*, first edition, June 2000. ISO/IEC 7498-2.

[26] International Organization for Standardization. *IT - Code of practice for information security management*, first edition, September 2001. ISO/IEC 17799.

[27] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*, November 1998. Request for Comment 2401, http://www.ietf.org/rfc/rfc2401.txt

[28] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*, September 1993. Request for Comment 1510, http://www.ietf.org/rfc/rfc1510.txt

[29] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*, February 1997. Request for Comment 2104, http://www.ietf.org/rfc/rfc2104.txt

[30] Othmar Kyas. *Sicherheit im Internet*. International Thompson Publishing, 1998.

[31] RSA Laboratories. *"PKCS #5 v2.0: Password-Based Cryptography Standard"*, March 1999. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/index.htm

[32] RSA Laboratories. *"PKCS #1 v2.1: RSA Cryptography Standard"*, January 2001. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.htm

[33] John Larmouth. *Understanding OSI*, chapter Writing it all down. International Thomson Computer Press, 1996. http://www.isi.salford.ac.uk/books/osi/all.html

[34] Peter Lipp, Johannes Farmer, Dieter Bratko, Wolfgang Platzer, and Andreas Sterbenz. *Sicherheit und Kryptographie in Java*. Addison Wesley, 2000.

[35] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. *SIP: Session Initiation Protocol*, March 1999. Request for Comment 2543, http://www.ietf.org/rfc/rfc2543.txt

[36] MIT Laboratory for Computer Science, RSA Data Security, Inc. *The MD4 Message-Digest Algorithm*, April 1992. Request for Comment: 1320, http://www.ietf.org/rfc/rfc1320.txt

[37] MIT Laboratory for Computer Science, RSA Data Security, Inc. *The MD5 Message-Digest Algorithm*, April 1992. Request for Comment 1321, http://www.ietf.org/rfc/rfc1321.txt

[38] National Institute of Standards and Technology. *Secure Hash Standard*, May 1993. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[39] National Institute of Standards and Technology. *Data Encryption Standard (DES)*, October 1999. http://www.itl.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[40] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*, January 2000. http://www.itl.nist.gov/fips/fips186-2/fips186-2.htm

[41] Tomas Olovsson. A Structured Approach to Computer Security. Technical Report No 122, Department of Computer Engineering Chalmers University of Technology, 1992. http://www.ce.chalmers.se/staff/ulfl/pubs/tr122to.pdf

[42] Lawrence C. Paulson. Inductive Analysis of the Internet Protocol TLS. Technical report, University of Cambridge, 2000. http://citeseer.nj.nec.com/1171.html

[43] Larry L. Peterson and Bruce S. Davies. *Computer Networks*. Morgan Kaufmann Publishers, second edition, 2000.

[44] J. Postel. *User Datagram Protocol*. USC Information Sciences Institute, August 1980. Request for Comment 768, http://www.ietf.org/rfc/rfc0768.txt

[45] B. Preneel, A. Bosselaers, and H. Dobbertin. The Cryptographic Hash Function RIPEMD-160. *RSA Laboratories' CryptoBytes*, Vol.3(No.2):pp. 9–14, 1997.

[46] J. Rosenberg and R. Shockey. The Session Initiation Protocol (SIP): A Key Component for Internet Telephony. *Computer Telephony*, 8, June 2000.

[47] Steven L. Shaffer and Alan R. Simon. *Network Security*. AP Professional, 1994.

[48] W. Richard Stevens. *TCP/IP Illustrated*. Addison-Wesley Professional Computing Series, 1999.

[49] Marianne Swanson and Barbara Guttman. *Principles for Securing IT systems*. National Institute of Standards and Technology, September 1996.

[50] D. Taylor. *Using SRP for TLS Authentication*, June 2001. Draft-ietf-tls-srp-01, http://www.ietf.org/internet-drafts/draft-ietf-tls-srp-01.txt

[51] Terry William. *Practical Firewalls*. Que Corporation, 2000.

[52] T. Wu. *The SRP Authentication and Key Exchange System*, September 2000. Request for Comment 2945, http://www.ietf.org/rfc/rfc2945.txt

[53] Study Group XI. *Q.931 - Digital Subscriber Signaling System No.1*. Telecommunication Standardization Sector, ITU-T, March 1993.

[54] Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman, and Deborah Russel. *Building Internet Firewalls*, chapter Security and the Internet. O'Reilly & Associates, 2000.